Interactive Image Processing demonstrations for the web

Author: Marcel Tella Amo
Advisors: Xavier Giró i Nieto
Albert Gil Moreno

Escola d'Enginyeria de Terrassa(E.E.T.)
Universitat Politècnica de Catalunya(U.P.C.)
Spring 2011

Summary

This diploma thesis aims to provide a framework for developing web applications for ImagePlus, the software develpment platform in C++ of the Image Processing Group of the Technical University of Catalonia (UPC). These web applications are to demonstrate the functionality of the image processing algorithms to any visitor to the group website. Developers are also benefited from this graphical user interface because they can easily create Graphical User Interfaces (GUIs) for the processing algorithms.

Further questions related to this project can be addressed to the e-mail addresses provided below[1]

1 Marcel Tella Amo: marceltella@gmail.com
Xavi Giró-i-Nieto: xavier.giro@upc.edu

Acknowledgments

It's time to say thank you to the most important people for me that has gave me energy to complete this project. First of all, to my tutor Xavi Giró, for the good organization for controlling me every week. Albert Gil, from UPC's Image and Video Processing group who has taught me lots and lots of things about programming, and he has also been directing the project. Jordi Pont also helped me a lot fixing very quickly a couple of errors found in ImagePlus code, so thank you, Jordi.

I have to thank also to my friends David Bofill and Sergi Escardó who have helped me a lot emotionally, in special those mornings and afternoons having coffee and talking about pains and problems in our projects.

In this group I should add Marina, my sister which has been very supportive with my project and has invited me to have breakfast lots of times.

And at last but not least, my girlfriend Raquel, who has been very patient and also supportive with the thesis and has helped me in whatever I've needed.

Thank you very much people! I appreciate it!

And here it is, the result of four months of hard work.

Index

Introduction

The main goal in this project is to improve the way how image processing developers can test their algorithms, and show them to other people to demonstrate their performance.

One of the top requirements for this project was how to find a very public and easy way to show this algorithms. The more users can see and use the algorithms, the better. Nowadays, web technology is increasing more and more its potential by creating a lot of applications, search engines, social nets, etc. which are a big source of information. Thus, the demonstrator uses Internet as a distribution channel and web languages to create a graphical user interface.

Web pages are shown by processing the HTML code, which can be statically written or dynamically written with other web programming languages.
A new way to display image processing algorithms is shown by using a C++ web toolkit called Wt.

In this project, a novel method is used to generate dynamic HTML documents, because instead of traditional languages such as PHP or ASP, the same language used for image processing algorithms has been adopted: C++. This way, software developers in the group do not need to learn new programming languages because the developed software layer takes care of interpreting C++ commands to generate HTML pages. Our work has been built on an existing open source solution called Wt, a web toolkit inspired by the popular graphical environment provided by Qt.

The Wt web toolkit not only generates static HTML documents but is also capable of capturing the event generated by the user on the graphical interface. This way, a certain degree of interactivity is provided so that

users can modify the parameters of the image processing algorithms and see the results on their web browser.

The system architecture proposed by Wt is desktop based, not taking care who is the server and who the client, while web applications always work in the server-client scheme which leaves all the data storage and computational complexity on the server while the browser is the web client that displays the results and captures the user interaction. This architecture concentrates the powerful machinery on the server side while allowing the clients to be light-based, whether consumer laptops or even mobile devices. This is one of the current trends in application development, commonly referred as cloud computing. This architecture has become possible thanks to the increase of data rates supported by Internet networks as well as a standardization of the communication protocols and data formats involved.

Requirements

Image and Video Processing Group at the UPC publishes every year several scientific texts in journals and conferences. In addition to the results included in these publications, the group is also interested in providing an additional channel to show the performance of their image processing algorithms. Given the current possibilities of web technologies, the main requirement of this thesis is to improve the software platform to offer the possibility of creating Graphical User Interfaces in the web. The requirement was completely fullfiled and even more, it was achieved to port the existing tools developed in C++ to the web, which was a success because it is the same language that ImagePlus uses.

The requirements are structured from two different points of views as the profile of the final user of this project could be either a programmer or a visitor interested in testing image processing algorithms.

Programmer's requirements

For the programmer, there is a need to test the algorithms in a visual way. This thesis will give an easy way of showing the algorithm in a web page, with the possibility of changing some attributes in the web page layout. It is expected to work at real time, therefore, by the time the image is being processed, the web page should display it.

The idea is to provide the programmer an easy way to create new pages for demonstrating other algorithms, with different attributes in each one. The main challenge is that the web demonstrator works together with ImagePlus, the library from UPC's Image and Video Processing group. It's written in C++ and for this reason, it would be better to have our whole application programmed in C++.

Furthermore, C++ is the programming language normally used for the group developers. The developed solution should not require any additional effort for these developers in terms of learning new web languages. For this reason, the use of CSS style sheets, very popular in web development to define the distribution of the elements in the page, should be avoided.
There is also another important requirement. The framework to use must be an open source framework to be able to use it.

To sum up, the basic requirement from the developer's point of view is to be able to develop web pages using C++.

External users' requirements

On the other hand, there are still requirements for non-programmers. They should see the application in a web browser as simply as possible.

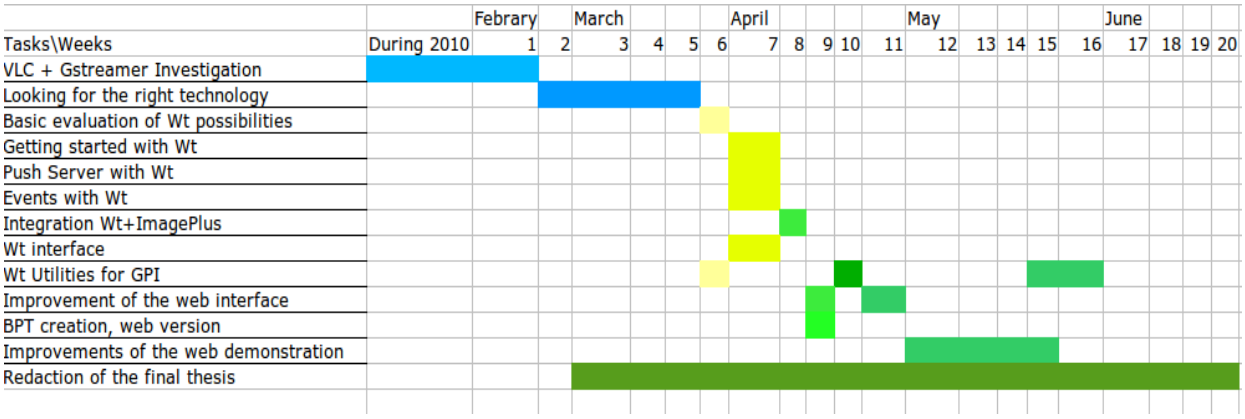
The website has to work without plug-ins and do not require intensive processing at the client computer.

It also has to work on the most popular browsers such as Firefox, Google Chrome, Opera, and even on Internet Explorer which usually is the most problematic browser in layout terms.

Work planning

Here there is a Gantt diagram of how has been all the whole time planning in the project. It is a bit confusing at the beginning because the project changed a bit from video streaming, video demonstrations and so on, to image demonstrations with data transmission.

The project has been building itself since a year before, during 2010 the project was started looking for the best research lines.



State of the Art

At this point basically will be explained how people show their algorithms as well as what technology is available to code web demonstrations.

Image Processing demos

Making videos

The most common way to show an image processing algorithm is by making a recording of the result given by the algorithm. This video can be published on Internet on sites such as Youtube or Vimeo.
There are a lot of demonstrations of image processing algorithms in You tube, I'm going to put some examples:

Terrassa TSC's youtube channel[2]

Silhouette Collisions Processing Demo[3]

Face detection and funny eyes[4]

Sorting algorithm[5]

University of California, (San Diego)[6]

They also have got some videos of each algorithm in the explanation in which they show how it progresses.

2 http://www.youtube.com/user/terrassatsc

3http://www.youtube.com/watch?v=rOgX23plbfg

4http://www.youtube.com/watch?v=9WXLs4qQ5nk&feature=related

5http://videoprocessing.ucsd.edu/~NatanHaim/scale_aware_saliency/comparison_touchdown.avi

6http://www.youtube.com/watch?v=Zk9WYO33k6w

Figure 1: GAT

Figure 2: Video demonstration

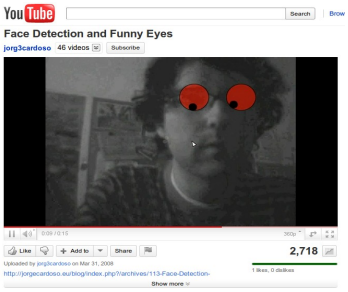


Figure 3: Funny eyes, video of the algorithm

Making videos and explaining them

This sort of demonstrations are both explicative and graphical, so you have an explanation embed in the video about what is the algorithm doing, what is the purpose of that algorithm, and of course, the video shows the progress of the video algorithm.

Here there are some examples:

- ***Segmenting Video Into Classes of Algorithm-Suitability***[7]

- ***Australian National University***[8]

They have got the video included in the main group's web page.

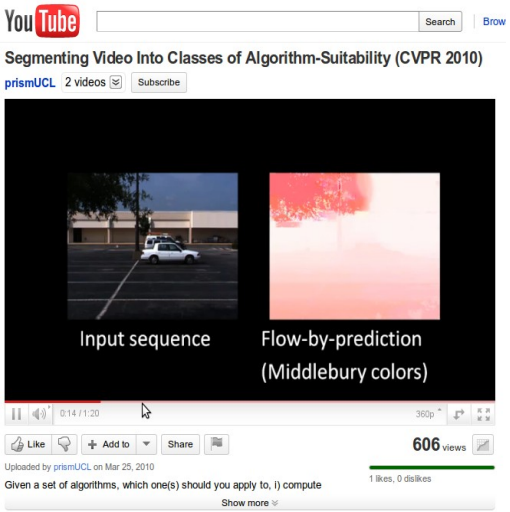


Figure 4: Explanation in a video

Figure 5: Video embbed in the webpage

7 http://www.youtube.com/watch?v=Zk9WYO33k6w

8 http://cvs.anu.edu.au/bioroboticvision/helicopter/hover_ext.mpg

30 Days trial

There are some companies and entities which let people try their algorithms by offering 30 days trial version in its web page. Usually it is done when it's a complete software rather than just an algorithm.

- ***NeuroTechnology***[9]

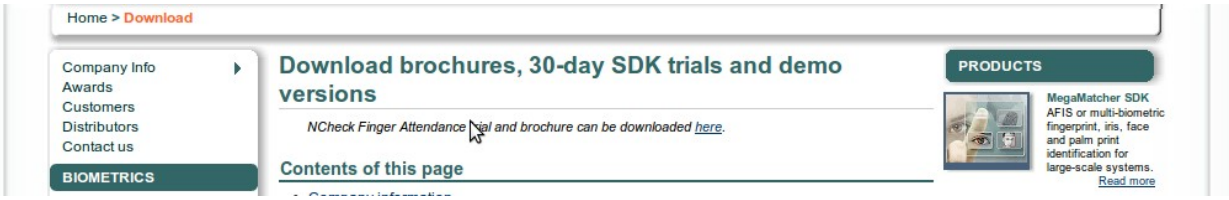


Figure 6: 30 days trial

- ***Colorado State University***[10]

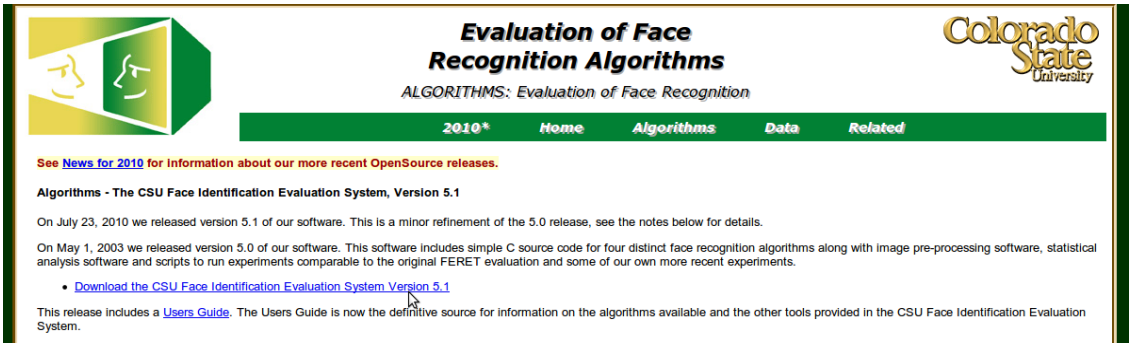


Figure 7: Evaluation software

9 http://www.neurotechnology.com/download.html

10 http://www.cs.colostate.edu/evalfacerec/algorithms5.php

Explanation step by step including the best pictures

There are some entities whose choice has been to explain the algorithm step by step in the same research group web page.

•***University of California, (San Diego)***[11]

is one example of this web pages, and you can find an example right here:

•***University of Hawaii*** [12]

They have also chosen to explain their algorithms in their web page.

•***RWTH Aachen University*** [13]

From Germany also uses text and pictures.

• ***Columbia University***[14]

Offers the possibility of downloading a powerpoint presentation of the algorithm, downloading some videos or reading there what it does.

•***UCLA Image Processing Research Group***[15]

They show all their algorithms as pictures of the results, adding a plain paragraph of information.

11 http://videoprocessing.ucsd.edu/~NatanHaim/research_saliency.html

12 http://www.ee.hawaii.edu/~treed/ispg/2D_seg_demo/demo.html

13 http://www-i6.informatik.rwth-aachen.de/web/Research/index.html

14 http://www.cs.columbia.edu/~allen/device/

15 http://www.math.ucla.edu/~imagers/htmls/inp.html

Publishing the source code

•*Bogaziçi University*[16]

Bogaziçi University's choice is to link the source code in the main web page.

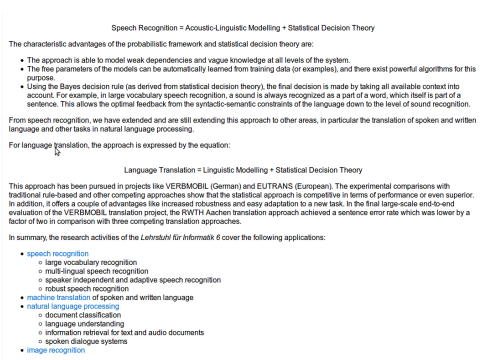


Figure 9: Text and pictures

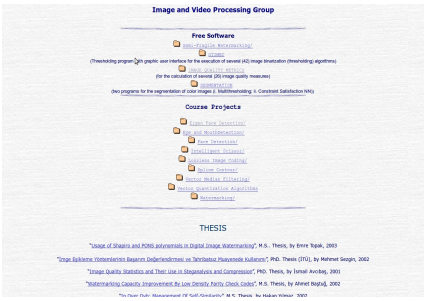


Figure 8: Source code in the net to download

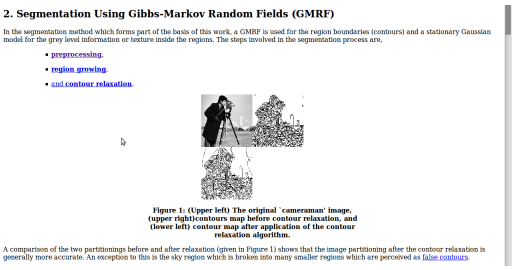


Figure 11: Graphics + Explanations

Figure 10: Columbia University, some possibilities to show algorithms

16 http://www.busim.ee.boun.edu.tr/image/Image_and_Video.html

Image processing algorithm in the web

At this point, probably the most actual demonstrator just based on images, it consist in a image processing.

- ***Carnegie Mellon University***[17]

Carnegie Mellon University has already got a system in which you can upload or search an image, send it and the algorithm processes that image and return the processed one.

This system is still difficult to use and it has some limitations as well as the size of the image, in this case is limited. The image that you upload has to have less than 900 pixels in both directions.

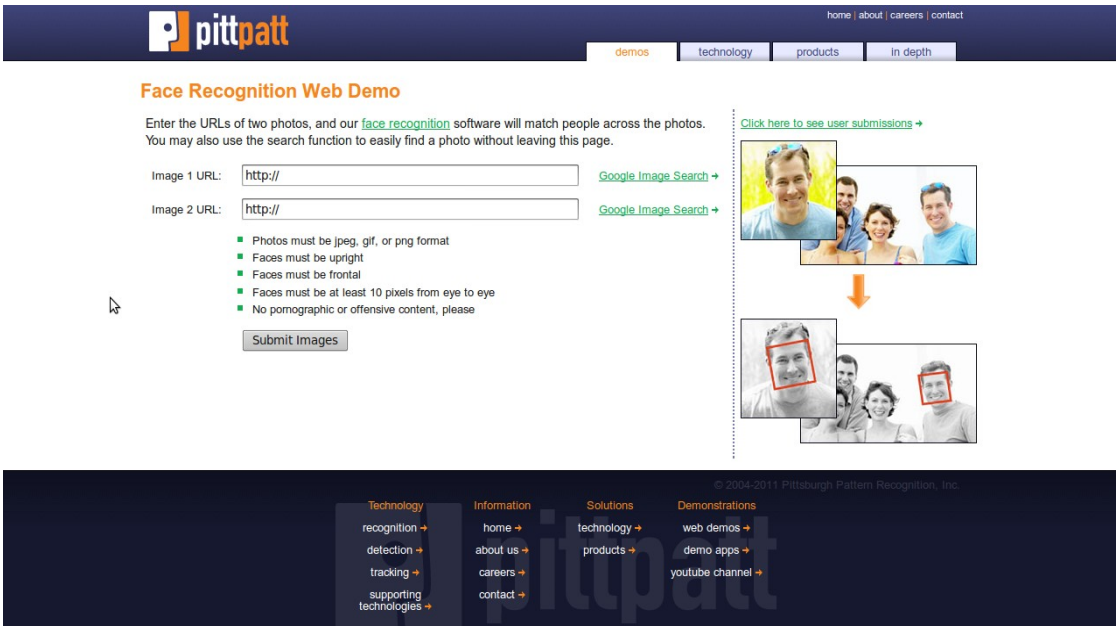


Figure 12: Image processing system

17 http://demo.pittpatt.com/recognition_demo/

- ***UPSeek***

UPSeek is another platform for the online access to the image processing algorithms developed by the GPI. It is divided in two parts: a server and a suit of clients, all of them implemented in Java. At the server side, UPSeek provides a Java wrapper for the C/C++ tools developed at the GPI that allows its online execution as web services. These web services use JSON messages and HTTP connection, both in GET and POST mode. The server is accessed both by the clients developed in the same project (an image ingester, searcher and annotator) as well as from clients developed by third-parties.

Open Source Frameworks and web technology

HTML

Hypertext Mark-up Language is the standard representation format in the web, all web pages are written, statically or dynamically in this language at the end.

CSS

Some time ago, HTML was useful to layout a webpage, using tables and other things to simulate a layout. For example, the <b> tag, which means bold.

The next thought was that HTML is just a representation language, but nothing related with the style. Since then, Cascading Style Sheets(CSS) were created, which were and they actually are used to apply styles and to layout a web page. Currently the latest version is called CSS 3.

JavaScript

JavaScript is an interpreted language in the client side, it means that all the code written in this language will be processed by the client's CPU. Currently is very used in the web although it can be disabled in every browser.

PHP

PHP, in contrast with JavaScript is a server side programming language. It's also interpreted so compilation is not needed.

AJAX

AJAX is not a programming language, it is just an architecture based on three different technology. One of them, the most important is JavaScript, and then a server side technology is requiered (PHP,ASP.net, etc) as well as a representation format for the data between them, which can be plain text, xml or some other formats.

Really, it means Asynchronous Javascript and XML, but XML cannot be used.

The principle of this architecture is that is not necessary to refresh all the page if there is just a slightly change. It is better to refresh just the modified part.

Actually we can see AJAX in a lot of web pages.

AJAX Push Engine framework (APE)[18]

Figure 13: Ajax Push Engine

APE is an open source AJAX framework for real-time data streaming. It includes both server and client libraries based on C and JavaScript respectively. What is more, it does not need any extra plug-in in the client side. It just works whether you have the JavaScript option enabled in your browser.

To sum up, as before explained AJAX allows you to query the server and get a response without refreshing the web page. You can see it as another block which is between the server and the client, which manage to request and retrieve data just downloading a little amount of information.

APE is a push-based system. This means it is not used for video on demand services, it's just for live systems.
The usability in the browsers is very important for all the web services,

18 http://www.ape-project.org/

because if an application does not work in a browser, people who use this browser will not be able to use this application. For this reason, APE was designed to provide full compatibility with all browsers.

Two different parts compound the whole system. The first one is the HTTP streaming server, which works with the APE protocol. The other one is the client side, which like I've said, a JavaScript framework is available.

Figure 14: Main Architecture of systems working with APE

There is a funny explanation in comic format, which explains that the main goal of this system, is to hold a connection between the server and the client instead the need of the client requesting the server. The server just starts the connection, and this connection remains opened during all the streaming time.

XmoovStream

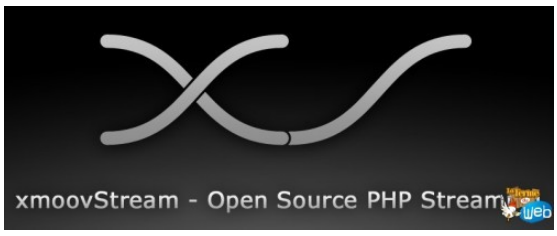


Figure 15: XMoovStream, a PHP streaming server

XmoovStream is a PHP based open source HTTP streaming server. It allows you to manage a lot of contents giving you control over many of them, in order to show an HTTP streaming in a web page. It has a FLV video and audio players.

This project is open source and free, in the web page you can get the source code, but it looks a bit difficult to get started and involved in the main architecture and functionality.

Nginx[19]

Figure 16: HTTP server Nginx

Nginx is a free, open-source, high-performance HTTP server and reverse proxy.

Clement Nedelcu has written a book about Nginx HTTP server, providing tutorials step by step about how to use it.

Boost.Asio library[20]

Figure 17: Boost libraries

Is a low-level C++ framework for network to provide a consistent asynchronous model of events. It also has a lot of functionality to create network elements such an HTTP server or an HTTP client, through TPC or UDP.

19 http://wiki.nginx.org

20 http://www.boost.org/doc/libs/1_42_0/doc/html/boost_asio.html

Libwww[21]

This library is a C++ Open Source library with the support of W3C. The purpose of libwww is to provide an environment for experimenting with extensions and new features. It is a high performance library to manage to do all kind of different things with HTTP and URIs. Library has been part of the World Wide Web from the beginning.

LibCurl[22]

Figure 18: LibCurl, a network library written in C++

Lib curl is a net framework which is designed to be very easy to use in the client side, handling protocols such as HTTP, FTP, POP3,RTMP,SMTP and a lot of different net protocols.

It has a very impressive quantity of bindings to write its code in the programming language you want.

C++ Network Library[23]

C++ Network Library is a project that tries to conform a high performance cross-platform library of networking, to interact with Boost C++ libraries.

In addition, this project is completely free and open source.

Wt[24]

21 http://www.w3.org/Library/

22 http://curl.haxx.se/libcurl/

23 http://sourceforge.net/projects/cpp-netlib/develop

24 http://www.webtoolkit.eu/wt

Figure 19: Wt, a C++ toolkit for web rendering,

Wt is a C++ library for Web programming purposes. As well as languages like PHP or ASP.NET, Wt works as the server side, but with the difference that C++ is a compiled language.

The main goal is to allow C and C++ programmers without any knowledge of web technology to make web pages and web applications without extra learning effort.

Wt has very good features like, like an own HTTP server which works when you start the program. It can also works with a common web server, for example, Apache.

HTML 5[25] new elements are included in the library, so you can display a video or paint some stuff in a canvas just with some methods of the library.

Furthermore, push features are included in Wt using different methods such as **long polling** or **web sockets**.

Urdl[26]

Urdl is a cross-platform for downloading web content, just using a URI. It provides a simple way to use standard C++ iostreams and an asynchronous interface. This library is copyrighted by Christopher M. Kohlhoff

Poco C++[27]

25 http://www.w3.org/TR/2011/WD-html5-diff-20110405/

26 http://think-async.com/Urdl/doc/html/index.html

27 http://pocoproject.org/

Figure 20: POCO libraries, Ac++ network framework

Poco C++ are a set of C++ network libraries which allows you to build network applications as well as interned based applications which run in your desktop.

It is an open source project, that claims to be used in industrial applications.

Design

Desktop vs Web application

The main goal of my project is to offer public and global demonstrations of the image processing algorithms of the GPI group.

Nowadays, a very high percentage of people in the world have got internet connection. For this reason, the most universal way to show these demonstrations is using Internet. This chapter describes the design process followed before reaching the solution that has been finally developed.

Certainly, there are other ways to show the demonstrations, locally. It could be to do a whole algorithm demonstrator software as a **desktop application**, and then to distribute it around the world.

There are few differences between this two types of architecture and each one has advantages and disadvantages. But the new trends tend to go through Internet, to the cloud. The most important business companies in the world of IT, such as Google, Microsoft, etc. as well as a lot more of companies are doing things to exchange desktop applications by web applications.The main advantage of the application online is that it doesn't depend on your operative system, but in contrast, it is necessary to have internet connection.

If a image or video Processing is executed in a personal computer, it strongly depends on the power of the pc, on the other hand, if the same application is in the net, it just depends on the power of the server or servers.

Another example is the company which was created in a village near mine, “Olesa de Montserrat”, this company is called eyeOS, it's an operative system in the cloud.

Actually there is an inconvenient in doing web demonstration, this is that if the program developers do fails for any reason, the whole server falls down. It means then the server is not accessible from outside, so, someone ought to turn it on again to be accessible.

To solve this inconvenience, there are a few solutions. One of them is to tell the server to send a message saying: “I'm alive” in this case, all is working properly. If this doesn't happen, it means that the server is off, then the server should be turned on.

The other way is to try to catch the exceptions produced by the program. But I think that is not fair enough, both are needed if a good performance is wanted.

Nevertheless, the demonstrations in this case could be watched by everyone with Internet connection, which is the very strong point of doing this.

System Architecture

Client - server architecture is the main scheme used by Internet to communicate itself between systems.

Its functionality is very simple. Considering that we have got two machines in the net, one of them is the server and the other one is the client.

It's similar to the meaning like when you go to a bar. In this analogy, you will be the client, and the waiter will be the server.

The client will ask for drink, so it will send a request to the waiter.

Once requested, the server just respond with an answer, it could be the expected data, or another kind of messages which will be explained now.

And how do client and server understand each other? If the waiter does not speak the customer's language, he will not be able to satisfy the request.

The technical answer to this question is HTTP protocol. This protocol is used between client and server to be able to send data, for instance a web page through the web.

HTTP is a **oriented connection** protocol. It means it needs a handshake of two messages, the request, and the response from the server. It's also a **stateless**, it does not change depending on the input, like for example TCP which does it. These changes are called “states”.

The request from the client

The request from the client is a message sent by the client to demand some data, normally in the web, a web page. In this example, typically the request is a GET request, but there are a few more.

Here there are the most important types of request.

GET

In the GET method, the information travels within the URI of the web page. In this method you are requesting both header and body of the data. It's not secure at all because the information goes through the URI, so you can see what are requesting for.

HEAD

HEAD method is very similar to get, but the only difference is that you are requesting just for the header, not the body or the content of the either the web page or data.

POST

The data travels now in the body of the message, commonly used in forms. The difference between get is that you can set more attributes provided that GET method has a limited number of attributes.

The main achievement in this point using the created framewok, has been to allow developers to program a web without taking care about net issues, just programming the application as well as if a GUI it were.

The response from the server

Once the client has requested, the server responds with other message. If it' OK, it sends an HTTP response with the associated number 200 which means that all worked fine, and the letters OK.

If it does not work fine, the server retrieves other kind of messages such as "404 page not found" or more messages.

You can see all sort of messages given by the server in the footnote in this page.[28]

Despite this, the most common responses are:

200 OK

It means that all worked fine and the data is correctly served.

403 FORBIDDEN

It means that you are not allowed to access to this data, so your access is forbidden.

404 NOT FOUND

This happens when you are looking for a page and you mistake a letter and the page does not exists then. The server has not found the page.

28 http://www.iana.org/assignments/http-status-codes

Structure of web applications

The algorithm demonstrator web page is supposed to be a page with all typical elements of an HTML document, but with some demonstration inside.

Web pages are normally divided in sections, so the algorithm demonstrator is also divided in these sections.

First of all, the algorithm demonstration is an **application**. It is an application that dynamically generates web pages to be shown in the browser.

Really, there are two parts, the server and the client, but what Wt wants is to treat it as a desktop application where there are neither client nor server.

Inside this page, we can find three more things.

The first one, the header, usually the header contains the information like the logo of the institution, some phrases and a menu.

The proposed design does not contain a menu. It's just containing a **logo,** a **phrase** or title of my web page, and also the **right space** for inserting some graphic design elements to make it a bit more beautiful.

The **content** is the most important part of the web page, because the useful information in all the cases is there, in this case, web algorithms are inside there.

Finally, there is the **footer**, the footer is where the copyright goes. Normally goes a site map also, the name and logo of the entity and the contact information.Joining all this parts, a whole web page is got to hold ImagePlus web demonstrations.

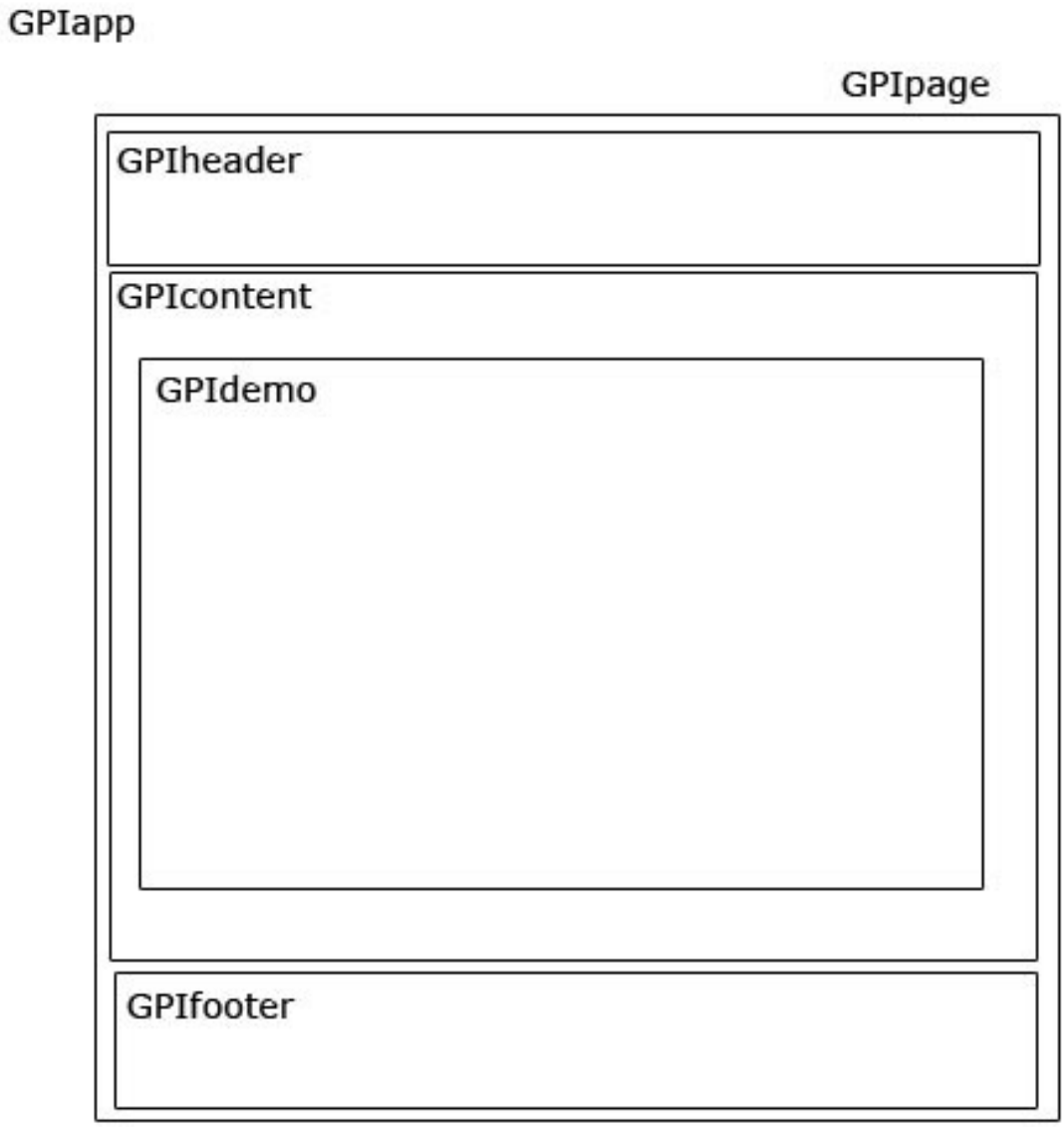


Illustration 1: Structure of the Class

Illustration 2: Hierarchy of classes

Integration of Wt in the ImagePlus platform

Libraries are always required when programming, in a local machine is not very difficult to link with the libraries, but it was done, so in the Image and Video processing server where the project was made via remote machine.

First of all, a lot of investigation had to be done in how Wt worked.

There were two important libraries to link, the first one was called “libwt” and the other one “libwthttp”.

These provide all Wt functionality and also an HTTP server, so there was no need to use even a Apache server, it's an own HTTP server from Wt.

WrasterImage, an element from Wt, uses functions from other libraries so we also had to install the library “grahicsMagic”.

It was possible to code and compile it by doing “scons -j4 project” which was automatically set in eclipse by just clicking a button.

After that, Albert decided that, well It is good that web demonstrations are like tools, but they are “web” demonstrations, they should be in something like “web” tools. So he created a directory called web.

And in the end, was added a variable called USE_WT, it is an option the user can set when compiling the program. And this is because is not necessary to compile and link with Wt is the program is not depending on Wt.

Webtools

On ImagePlus there are “tools” which are the little executable programs, it means that every tool has got a main function.

There was a discussion about the idea of doing web demonstrations in the same way that if they were an ImagePlus tool.

The other possibility was quite interesting, and perhaps this possibility does not have to be discarded for the future. It consists on a web page with a demo and a menu, where the user can browser between demonstrations.

On the one hand, by doing single demonstrations could be very useful for developers because when some wants to try algorithms, it is needed to be seen fast, with less traffic.

On the other hand, it is more comfortable for the visitors to have a single web page in which you can navigate and see all the demonstrations.

Fortunately, an easy alternative will be to install a copy of the demonstrator in a local server, and then the developer could see their application at the moment, and in a web environment.

To summarize, to do single web demonstrations has been chosen, to help developers even with the graphically debug their applications.

Resources

All elements needed like images or other things are also part of a web page, this is what is called “resources”.

The management of the resources is not so easy. There are resources which are shared by all demonstrations, such as the logo, or the background images.

This resources should not be copied for every demo but shared from a common directory.

And if you are running the program from the command line. Every web page has a root where all the file hierarchy starts.

Actually you specify the root in the command you type to execute the application.

```
--doc-root /location/of/the/root
```

Interactive Elements

The application is supposed to have interactive elements. Developers and external users should be able to change some parameters of the algorithms in an interactive way.

The matter is how to do it, Wt have exactly the same philosophy of Qt, and also the same function names. The technique related with evens is called Signal – Slot.

Almost every element in Wt can be connected with a signal, when an event occurs, then a signal is thrown. Once the signal is thrown, it has to be a slot which is a callback function that is called every time the event occurs.

In this manner you can set "event listeners" by connecting signals.

Here there is a very easy example to understand how it works.

```
Element->clicked().connect(this, &Class::CallbackFunction);
```

This is a real example of how should a signal be connected.

At first it is necessary to specify the element which is going to throw the event.

```
Element
```

Using pointers, there is a → sign after it, to access to the mehod "clicked()".

There is a callback function which is the function to execute when the event starts. The Class is also mandatory, it has to be the same class that it is being built at the moment.

As said, Wt uses the same method that Qt, for this reason, it is good for the integration in the software platform. It is actually a known technique for programmers there.

Trials and demos

The main dificult of the project was to search in the great pool of web technologies which one, or which combination of them, is the best to futfil the requeriments. So a lot of little test have been done to compare are discard technologies.

Some important features were tested for web demonstrations.

Push

Push is just the ability the server has to send messages without being requested by the client, it just keep sending messages whenever it wants.

Ajax Push Engine: As the name say, it's a framework thought to push data, its mainly used by game developers.

The main inconvenient of this framework for this case is that it is programmed in Javascript, so it would be difficult to integrate, it should to be easy to integrate with the development platform ImagePlus.

Wt: It just allows you to do this and it is not very difficult, there is an example in Wt's documentation explaining it.

Painting images

Some things were tried before getting a stable solution. The first trial was to use the HTML 5 canvas element combined with Javascript to to paint something there. A clock was painted and it worked perfectly.

But the main problem of this system is the same than before, it's JavaScript again, finally C++ was mandatory to connect all the elements.

After trying this, the goal was start trying Wt, looking for methods to paint an image.

The surprise was that Wt already had a type “Image” with an element to change every pixel.

It was a bit hard to discover but finally it was possible to paint an image in the web.

But it was so difficult then, and it did not have any sense for the project to have two elements, the image and the resource separately because all wanted was to show images, the image processing was and currently is done by **ImagePlus**.

The class was made and it is called WebImage. We will see how it is implemented in the next chapter, results.

At first, with so many requirements, it was not clear what technology was to be adopted. Some web technologies were tested, preparing little demonstrations for each one to see if it could fit in the whole project.

Painting in the HTML5 canvas

The first idea was to display a dynamic image in the web, was the <canvas> tag recently included in HTML, with the last version HTML 5.

This tag allows you to "paint" over a sensitive zone in the web page.

A little toy man was done, with needles in its hands, symbolizing a clock with the hands. It was developed with a JavaScript framework called **Liquid Canvas.**[29]

In this demo, there were two main goals, first of all, paint something in a web page, and in second place, send data from the server (the time).

29 http://www.ruzee.com/files/liquid-canvas/demo.html

APE (Ajax Push Engine)

The first impression of APE was very good. It has very good design and some impressive features, It's mainly thought to develop games online interacting with a lot of players.

APE has an easy way to do data streaming which is good but It had a thing that make me change my opinion.

In the client side it is programmed with JavaScript, and rarely in the server side is also programmed in the same way.

JavaScript in the server side didn't help the integration with ImagePlus at all, as the server must run C++ image processing libraries.

Using CGI to test C++ scripts

After the demo above showed, was thought that JavaScript was not a good alternative because of the programming way of the UPC Image Processing Group. They work in C++ and the main difficulty would be to find the way to join technologies. So was better to find directly some library or some way to execute C++ scripts in the server.

In this case, was made another demo. It was an input field where you could fill it in and a different message was shown depending on the input.

Exploring different libraries available

C++ was the clear option, for sure. Investigation to look for some library which was able to cover all my requirements had to be done. I was investigating some of them and really there are very good libraries for streaming and net purposes.

I you can see some examples in **the State of the Art.**

Finally, Wt was chosen because of a lot of useful features which will be explained in following chapters.

Testing Wt, first web demonstration

The first web demonstration was just based in a Wt example, which just said "hello" in a web page.

This demo was just to test how Wt could generate HTML and JavaScript interfaces internally, just using C++.

And in fact, it's possible and quite simple.

Pushing data and painting in Wt

This was a very good advance in the project. A very little application but not easy was also built, the server sent random coordinates whenever it wanted, and in the browser you could see these coordinates like little circles.

It is really a very good feature because nothing is required in the client to receive data every time. This is push.

The painting issue was good to make myself sure that I could display an image dynamically with Wt.

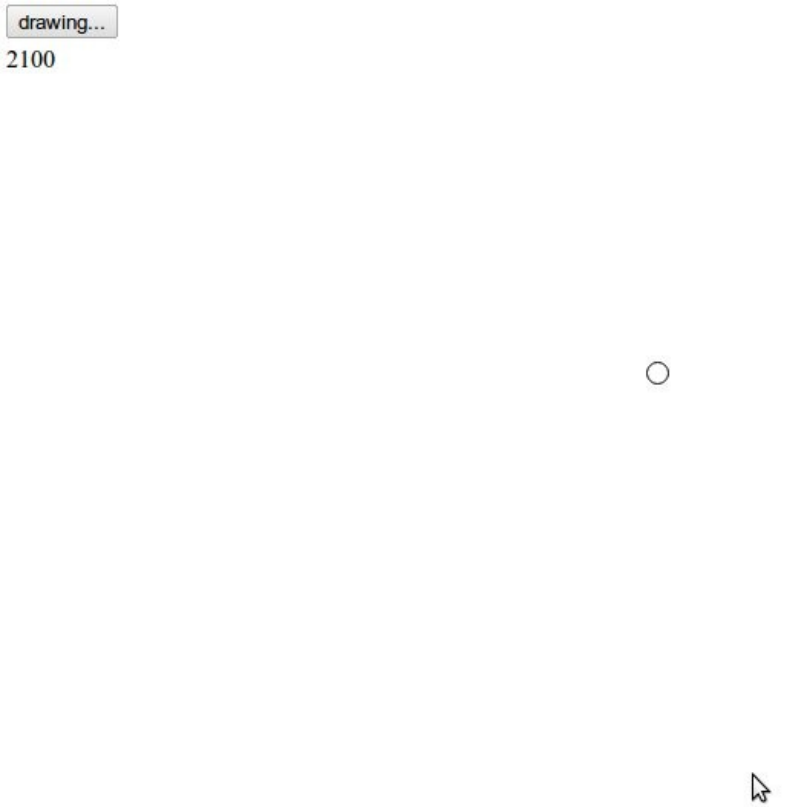


Figure 21: Wt Push Server

Events in Wt

Some more examples were done to test whether Wt was useful to interact with the user or not. Finally, it was discovered that it is possible to achieve, and it is not very difficult. The philosophy is "signal and slot". You have to throw a signal and in the other part you should do a callback function which is called once the signal is thrown, like for example, when you click somewhere or you move the wheel.

Using the wheel

Talking about this last concept, another illustrative example to test the wheel was done, if the user moves the wheel up, it sums 1, and if the wheel is moved down, it subtracts 1. The output was just a number.

ImagePlus with Wt

My first demonstration in this way was to calculate a Scalar Product between two vectors entered by input fields.

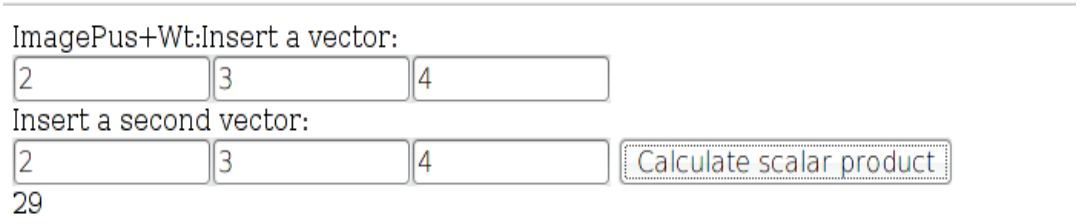


Figure 22: First try ImagePlus+Wt

Generating Images from ImagePlus

The next was to generate some plain images with ImagePlus. To add a bit of beauty to that issue, it was decided to paint 4 colours and to divide a square in four parts. When you moved the mouse over the first square, the colour was one, then, when you moved the mouse over other squares the colour was changed.

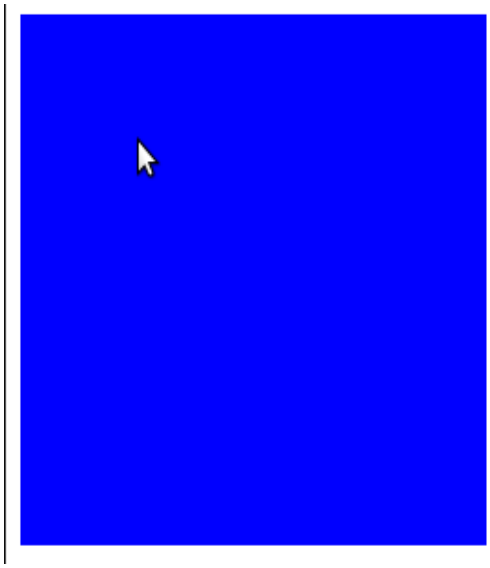


Figure 23: Interactive demo, events

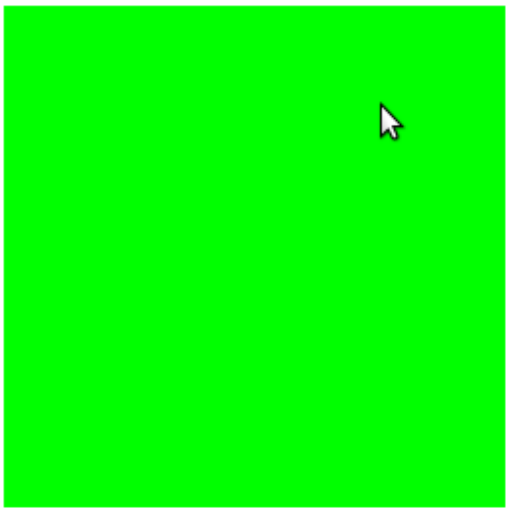


Figure 24: Interactive demo, events

More events to Low level, average of pixels

Now, it was time to down to pixel level. The goal of this example is to be able to do some calculations with the pixels. This demonstration just looks the neighbor pixels and does the average for the three channels, red, green, and blue. As well as the total average.

Figure 25: Average of pixels demo

First web interface, just Wt

This was the starting point of being able to do easy web demonstrations. A whole architecture of a web demonstrator, like a web interface was built.

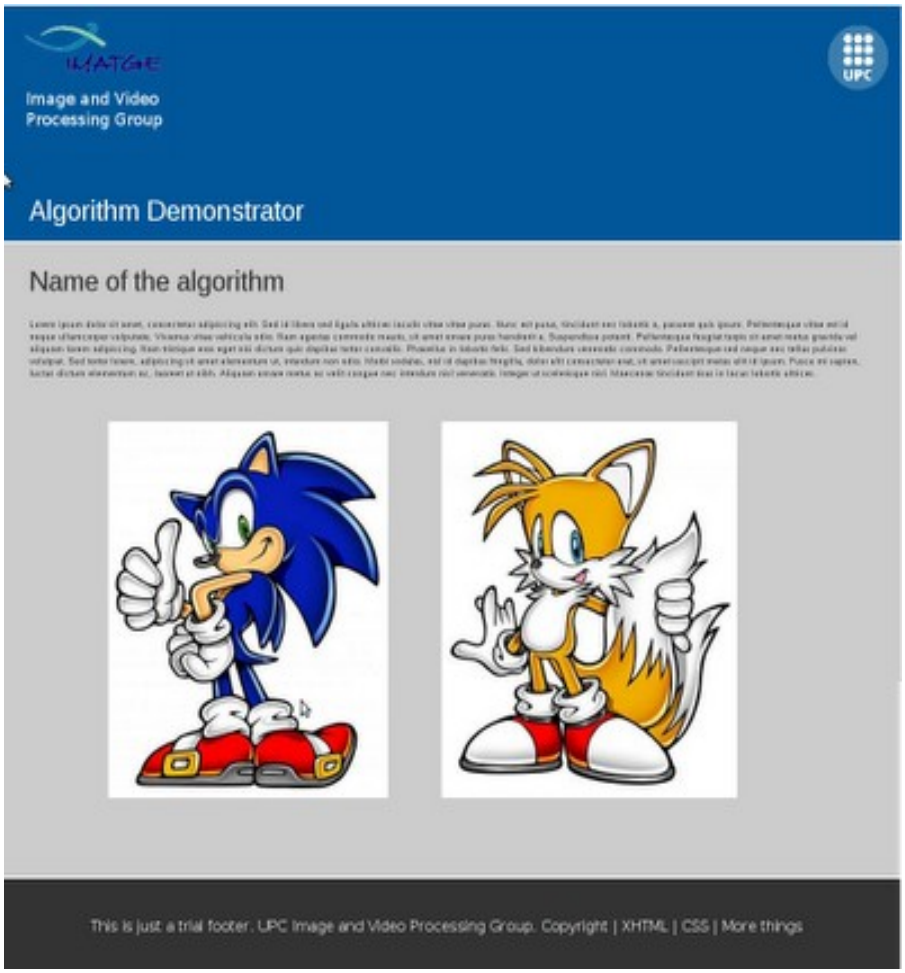


Figure 26: First web Interface, just Wt

Everything in this demo was displayed with C++(Wt), but really I programmed a bit of CSS to fit everything in place. This really does not affect to developers because when programming a demo, the developer will get the web interface immediately by inheriting from my class without taking care of CSS.

The architecture will be explained in following chapters.

Wt Layouts vs CSS

As I've said before, CSS is the web technology which aims in adding some style to a web page. But the requirements of the project clearly state that the solution must be completely based on C++.

For this reason, the investigation aimed to use layouts. Layouts are something like containers where you can put things inside, and they are displayed horizontally or vertically depending on which kind of layout is, if vertical or horizontal.

Layouts can be nested inside other layouts, so you can have a vertical layout and inside this, have an horizontal layout, and then you get a better distribution of the space.

This working method is very typical from Desktop environments, Qt uses it and it is what Wt is trying to do. It is trying to transform the programming method of desktop applications into web applications.

Development

Title and description for each web demonstration

To show the algorithm graphically is great, but also some other things are required in the interface.

These things are the title and a little description of the algorithm which is being showed.

Every developer **have to** set a title and a description of this web demonstration, if they do not do so, the whole application will report them a compilation error, so it is not going to be executed.

Taking a look at the interface, we can see than the GPIpage contains all the other containers.

When you inherit a class from GPIapp, the constructor of GPIapp is called, so we can, then put the title and description. In addition it is also called the constructor of our new class.

Then the solution is to pass the title and the description from the actual web demonstration to its parent class constructor.

The constructor then looks like this:

```
demo::demo(const WEnvironment& env):
 GPIapp(env,
         “This is the title of the web demonstration”,
         “And this could be a little explanation”
         )
```

Well, as you have seen, it's very easy to set the title and description of the web application. Despite this, there is a little explanation of every concept in these lines.

```
demo::demo(const WEnvironment& env)
```

This part means that we have got a class called “demo”, and that this function is a constructor because it has the same name the class has.

It has a parameter as argument. It is just information about the environment of the web page.

```
:GPIapp(env,
          “This is the title of the web demonstration”,
          “And this could be a little explanation”
          )
```

What this means is that in addition to call the class constructor, the class also calls its parent's constructor, and it just creates the title and the description.

Pointers vs Objects

In this project there was a big difficulty to face when talking about the style of coding.

Wt tries to do everything with pointers, like in C.

```
otherclass *i;

i = new otherclass(constructor);
```

But ImagePlus in the other side is currently using the C++ style which is using instances and objects. For example:

int i;

Programmers have already got a variable called i, you can do whatever you want with it, in this way you also could do:

```
otherclass i;
```

Even though pointers are also a very important part of C++, the improvement between C and C++ is supposed to be the facility to manage memory issues. It is not needed to use “delete” and “new” which are used to free and reserve memory respectively.

Naming convention

The final project is part of ImagePlus, it is a little “framework” which allows developers to create web demonstration in a very fast way.

So I had to read and learn the way how Image and Video processing group usually name the functions, the variables,etc.

This is the way how Wt names the functions.

```
ImAFunction();
```

This is the way how I have to name functions on ImagePlus

```
im_a_function();
```

Also, the name of the properties of the objects:

Wt:

```
int i_;
```

ImagePlus:

```
int _i;
```

The underscore is just to indicate that it belongs to a class. This way programmers can identify in very fast what it is.

In the foot of this page you have two links, the last one refers to the whole ImagePlus documentation, where is the convention naming from them.

It was an effort to get used to the Wt naming convention and then changing to the ImagePlus naming convention also was an extra effort which has been faced.

Layer between ImagePlus and Wt

There is indeed a layer between both technology, in the sense that some utilities have been created to easily combine ImagePlus with this layer, in case that Wt programming were difficult, or simply in case that had been considered that the utility is going to be used other times.

Once said this, here there are a few utilities made to make easier to program just knowing the minimum about Wt.

Image container

The summary of this point is that WebImage is an image that accepts an ImageRGB<uint8> from imageplus and paints it in the screen. When you instantiate this, you get an image rendered in the screen depending and what you have passed.

Anyway, I'm going to explain exactly how it is internally.

```
class WebImage
{
    private:
        //The <img> tag in HTML
        WImage _ima;

        //Where the image will be stored.
        WRasterImage* _Rima;
    public:
        //Constructor by default
       WebImage(WContainerWidget *parent=0);
        //Constructor by parameters
       //WebImage(ImageRGB<uint8> &input,WContainerWidget
*parent);

        //Get WImage
        inline WImage& getImage () { return _ima;}
        //Paint image
        void paintImage(const ImageRGB<uint8> &input);
};
```

As you can see, it just have two elements, a Wimage and a WRasterImage.

The constructor just construct the image in a way by default and then the image is painted by using the “paintImage” function.

The main facility that this class gives is that gets a ImageRGB passed as argument and copy it to the resource to be painted, but first it check if the WRasterImage has the same size as the image, if not, it is deleted and a new one is created with the correct size knowing that Wt does not have any method to resize a WRasterImage. This way I can avoid a lack of memory.

This is “paintImage”:

```
void WebImage::paintImage(const ImageRGB<uint8> &input)
{

    //Checking the size
    if (double(input.size_x() ) != _Rima->width().value() || double(input.size_y() ) != _Rima->height().value())
    {
        delete _Rima;
        _Rima = new WRasterImage("png",WLength(int(input.size_x())),WLength(int(input.size_y())));

        _ima.setResource(_Rima);
    }
```

```
    //Check the dimensions are the same.
  (double(input.size_x() ) == _Rima->width().value() &&
  double(input.size_y() ) == _Rima->height().value(), "DIMENSION
  MISMATCH");
   //if they are not the same, change the WrasterImage.
    for (int i=0;i<int(input.size_x());i++)
      {
          for (int j=0;j<int(input.size_y());j++)
          {

              //Assign the correct data.
          _Rima->setPixel(i,j,WColor(input(RED_CHANNEL)[i][j] ,
                                      input(GREEN_CHANNEL)[i][j],
                                     input(BLUE_CHANNEL)[i][j]
                                                      )
                                       );

          }
      }
      _Rima->WResource::setChanged();
}
```

There is a thing to remark and it is that in this way, the image rendered is going to be always in png format.

Results

Web Interface

As I've been remarking before, the main result of this project is the possibility that each programmer can do their web application in a very easy way.

A new web framework is included there, so everyone can use it.

I'm not really a designer but I've tried to do a formal interface and friendly at the same time, with the UPC colors.

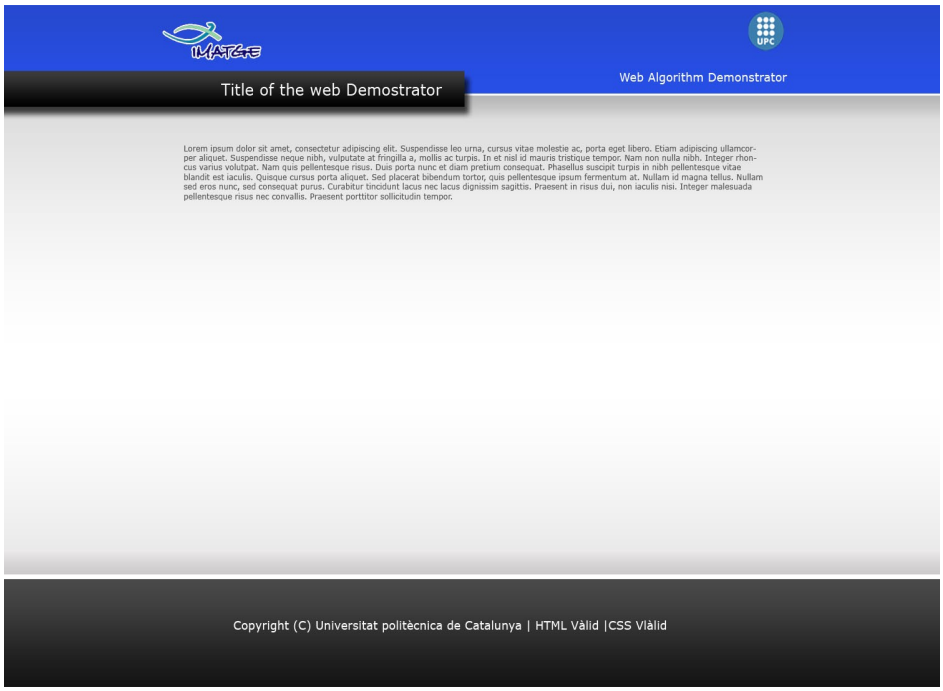


Figure 27: The whole interface, just void

Binary Partition Tree as web demonstration

Here starts the interesting part. Xavi and Albert suggested me to use the Binary Partition tree algorithm to make a web demonstration. They showed me what code I should use, as well as Jordi Pont, and I did the web demonstration from that algorithm.

Basically it works by moving an slider.

When the page started, a binary partition tree was created. It means, ImagePlus divides the image in the minimum unit and by joining little regions it finally ends in a whole tree of lots and lots of regions.

Then when you moved that slider, the average of colours of the leaves of that tree were shown graphically in the screen.

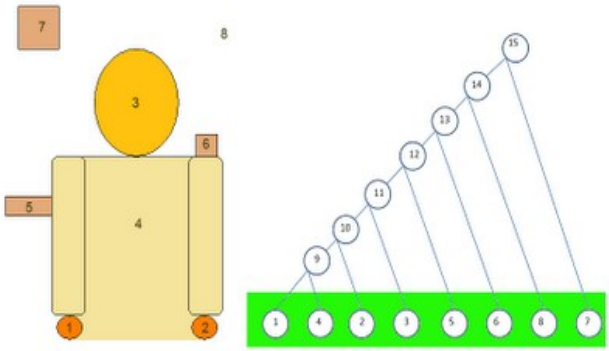


Figure 28: Binary Partition Tree

For more information about the BPT, look at the Lluís Garrido and Philippe Salembier's article.[30]This figure has been taken from Pia Muñoz's thesis. For more information, the URL is below.[31]

30 http://dx.doi.org/10.1109/83.841934

31 http://gps-tsc.upc.es/imatge/_Xgiro/teaching/thesis/2009-2010/PiaMunoz/memoria.pdf

Little introduction of the BPT algorithm

BPT means Binary Partition Tree. It is essentially a tree whose leaves are regions of a fine partition of an image. If you go to the top of the tree, you can see the whole image as a region, in case that you go to the bottom, you can see all the smallest regions in the image.

In this figure we can see a face divided in regions as well as the regions at the same time in more little regions, becoming a tree.

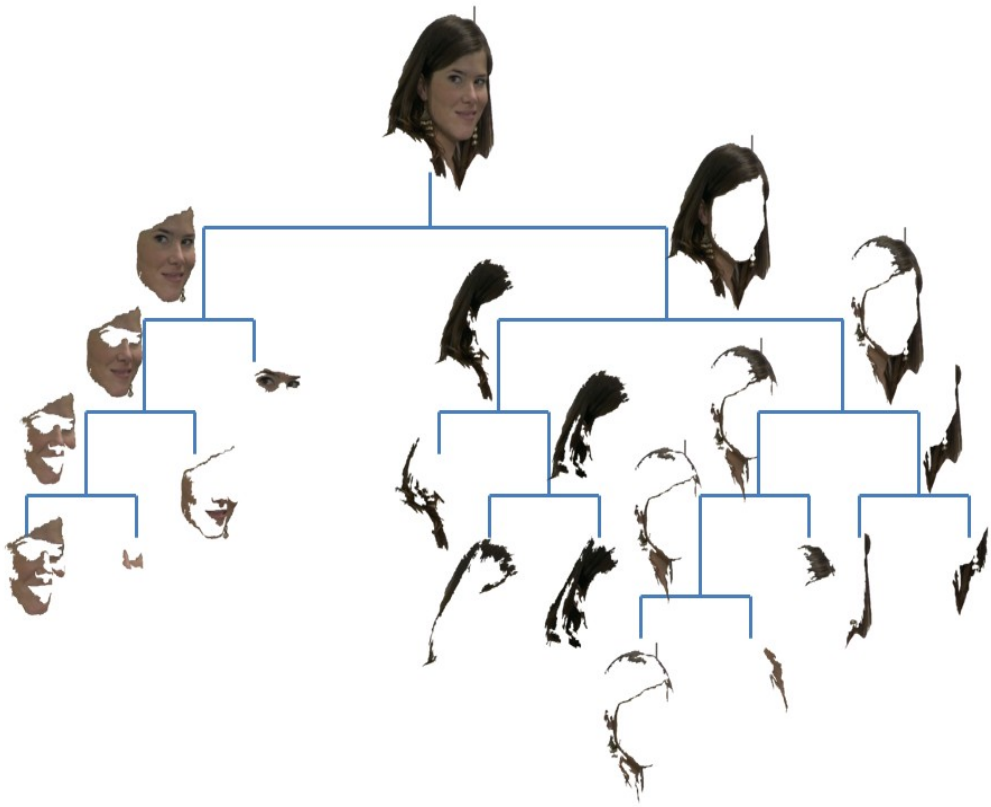


Figure 29: BPT creation tree example

Demo description

Once created the tree, we can navigate for it, paint the average of colour of this regions, overlay the borders, etc.

It was tried to do it interactive by putting some elements in the web demonstration.ut the main use of the tree in this case is, at first, it has to be created. Then when you interact with it, it just calculate the regions it has and paint the average colour of the region.

Can't you still imagine this? Take a look at the following figures.

Figure 30: Sonic bpt demonstration

Figure 31: Sonic bpt demonstration

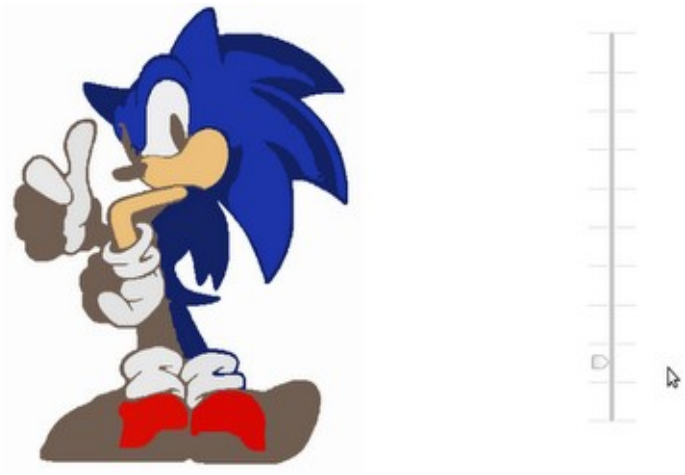


Figure 32: Sonic bpt demonstration

Improvement of the web interface

All web demos must show a common look & feel to give a sense of homogeneity among the demos generated within the group. This is achieved by making mandatory for developers to provide a title and a description for their demos. These strings are then used in the creation of the HTML document to provide a structured and predefined outer layout. These contents were formatted by adding some shadows and shapes.

In the same example, the demo was inserted.

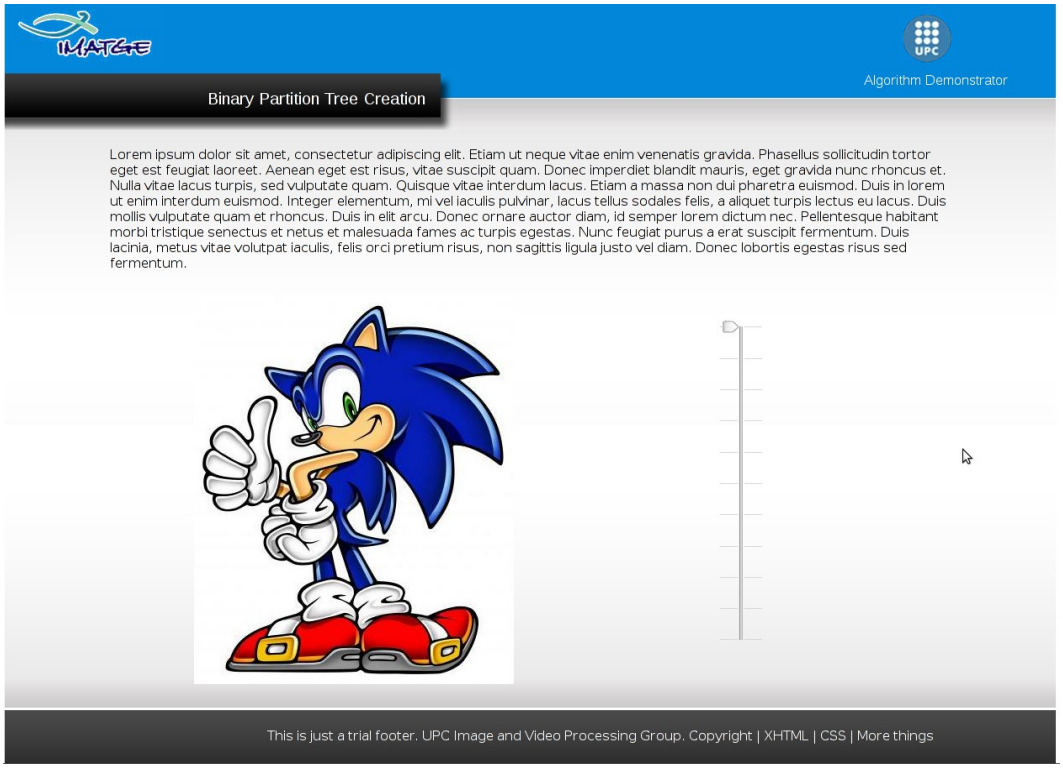


Figure 33: New web interface, improved design and features

And after this some improvements were done to this web demonstration.

Making a complete web demonstration

Some Wt code was written to add this utilities such as a File Uploader, two images from the class WebImage that will be explained later, a little label to identify the number of the region and a combo box to choose from images in a directory from the server, which automatically displays all the images in that directory.

The demo finally contains:

•**Image original:** A WebImage Class.Just shows how the image looks like.

•**Result of the processing:** Another WebImage which shows the result of the image processing.

•**File Upload element**: The way how you can put your pictures to see the demonstration with them.

I had a lot of problems with this because It just stores the uploaded image in a temporary directory. At the beginning it just didn't work.

I thought it was some kind of permissions stuff. But really it was not the problem. The problem was the file size which was allowed in the specification file of Wt.

Then, I discovered a method from Wt::FileUpload which is to detect if the file is too large, and if so, do something interesting with it.

•**Combobox:** It just looks at one directory and then charges all the images from this directory. Really it is not more useful than the file upload element, but it's another option to give the viewers, just work with images already in the server.

•**Button:** Just says “Set Borders” and draws the border of the regions in white

•**Id partition:** Is a text field which indicates the id partition you are clicking over.

•**Slider:** It's just used to go up or down in the tree(BPT). You can then see more regions or less regions by going up or down respectively.

•**Wheel:** Has the same efect than the slider but it's not necessary to move it, when you move the wheel, it just do the image processing and move the slide automatically as if you had moved it.

I faced a problem here, and it's that the web interface already has an scroll to go up and down moving the wheel, so I had to find out how to fix the conflict.

The solution was definitely disable the parent actions, letting thus just processing.

•**Loading element:** This element is very special because It just appears when the page is charging. But it's very strange when it is calculating, because it seems like the server is lazy . So I wanted to change the message to "Calculating...". This were also two lines of code.

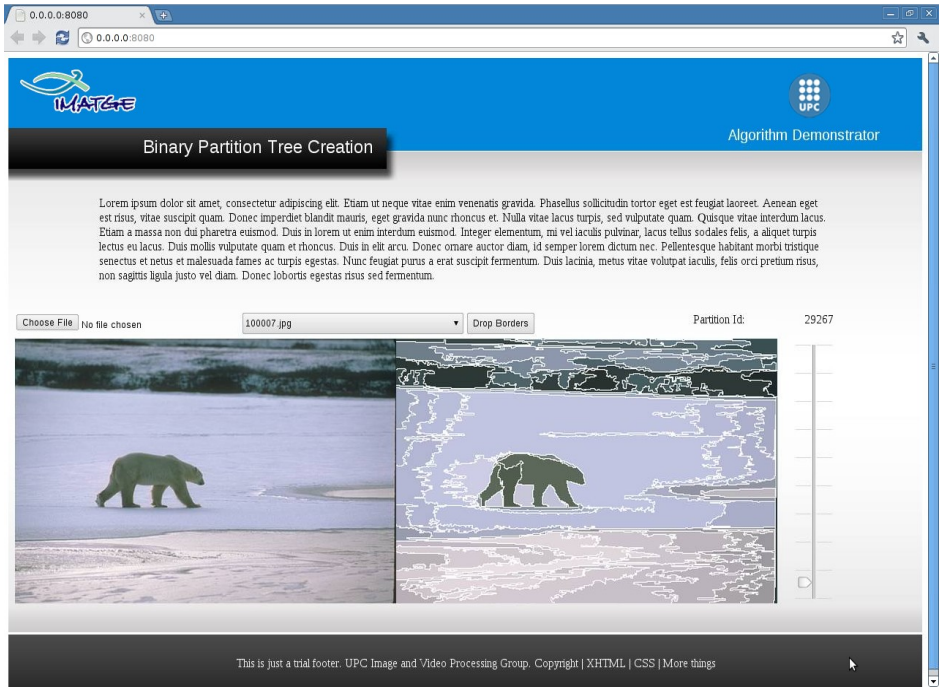


Figure 34: BPT creation, whole web demonstration

Showing bibliographic articles

The web interface will be useful to put demonstrations inside but almost always this demonstrations have been derived from articles from other people. And these articles have to be reflected there.

For this reason, a bibliographic class has been done to reflect these papers.

Really, it is not just one class, there are two classes.

The first one, “WebBibliography”. It is who take the text and format it in the correct way.

The second one is not too stylistic, it is part of the web interface, it is placed below the GPIcontent as brother, but inside GPIpage.

It allows the bibliography to be below the demonstration always even though the demo is too extense.

The second class is called “GPIbiblio” and it does a couple more of things. It puts the label “References:” if there are references, if not it is not set. Also, it controls how many references there are, and pass it to WebBibliography which is inside, to render a little number automatically.

Finally it looks like this:

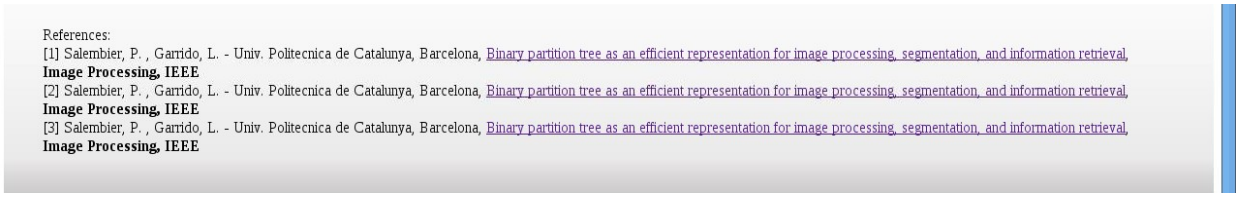


Figure 35: Web Bibliography widget

it is optional and there is a method in “GPIapp”, so to add a reference, the programmer just have to do:

```
addBibliography("author","urltitle”,"Publication","hyperlink");
```

Multiplatform testing

Once the application was done and working properly, the Image and Video processing group decided to give me one IP and port to run my application to the web.

By doing this allowed me to test my application in different browsers. Furthermore, I went to Belgium, concretely to Gent, which is very beautiful and I strongly recommend to visit it, I tried my application from there, and it worked correctly.

Also Xavi tested it from New York city. But as known, Internet is global, so the really important part is whether the application is multiplatform or not.

The server

It is very important to have a good server in these kind of projects. Because if the server has a poor amount of RAM, what happens is that when an application starts and it keeps moving images and some more data, it will run out of RAM in a fast way , and it will have to use SWAP memory, which is much more slower than RAM, and every loading in the page can stay around 40 minutes instead twenty seconds with the ram.

And this is taking account of that there is just one person connected, of course in a real net environment it is not going to work.

Documenting Getting Started

Every programming project requires a documentation and even more if it's a framework thought to be used by more users.

For this reason, and using **doxigen**, which is a system to generate HTML pages with documentation by documenting the source code.

A littie piece of documentation was done, which is a little tutorial involving the most important things to get started with my framework and with Wt.

Source Code Documentation

At this point, it's high time to comment the classes.

ImagePlus have a system called **doxigen** which, commenting the source code with some comments, you get a whole documentation in HTML, all formatted and quite good.

Well, basically this comments have to be in every function, every parameter, in, out or whatever.

The main tags I've used to comment it are:

`//!`	In all which goes in doxigen has to be placed
`\brief`	It is used to give a description
`\param name : description`	To identify and explain a parameter.
`\param[in] name : description`	To explain an in parameter.
`\param[out] name : decription`	To explain an out parameter.
`\return : description`	To explain the return value of a function.

Here there is a short example of the comments system.

```
//!
//! \file GPIapp.hpp
//!
//! \author Marcel Tella <marceltella@gmail.com>
//!
//! web framework
//!
```

```
 //!
 //! \brief Basic contructor.
 //!
 //! \param[in] env          : The environment var given by Wt
 //! \param[in] title        : Title which appears in the web
interface
 //! \param[in] desc         : Description of the web
demonstration
```

Users test

the success of this thesis depends on the adoption of the proposed technology by developers community at the GPI research group. For this reason, the developed framework has been made public to them to evaluate their reaction.

If they succeed in doing their demonstration, it means the documentation is understandable and easy to follow.

If they can't achieve their goal, it is a warning, to improve the documentation to make it easier.

Conclusions

Good way to demonstrate algorithms

As seen in the State of the art, there is not a clear trend of using Image Processing Groups in the web, as algorithm demonstrator.

So has been probably one of the first ways to do it, which is very good for me, and also for the UPC Image and Video Processing group, which has now a lot of facilities implemented to make their web applications, and besides, the web framework is though to be used by whoever want to use it because in the Image and Video Processing groups, before this project they did not even do GUIs.

Universal Access

Another powerful feature of the web demonstrator. It is “web” so in principle, it can be seen by all people around the world with a computer and an Internet connection.

The architecture of the web is client-server. And these two are the computers which work in the moment the application works.

Well, the other advantage the application has got is that the client is doing nothing while the server is doing all the hard work, processing etc.

This is great because an old machine with not too much memory can access the same way to the web page, and going further, there are other devices whose CPU is not very good but the application can run in them.

The future, mobile demonstrations

Nowadays, the number of applications programmed in order to run in mobile devices is increasing very fast.

Besides, mobile phones are already able to connect to internet, the application could run on them.

It has to be really well thought because is not trivial. Mobile phones used not to have the best processor in the world. I have to be aware of this situation, and in this case I can't allow the mobile to do all the operations, I should order the server to do all the operations and then the client just displays it.

Figure 36: Thinking in future applications for mobile phones.

Future Improvements of the web demonstration

Highlighting a leaf

What is also wanted is that when you click over a region, the group would like this region to be highlighted, and then moving the wheel, you can get the parents and sons of this leaf.

Finally you will be able to navigate through the Binary Partition Tree.

Selection marker

This was one thing in which I wasn't working alone. **Victor**, other student directed by **Ramon Morros** from the Image and Video Processing group was also doing.

It is not an easy issue because it have to interact over a image, and only over it. Well after that, a square has to be painted at real time, so if we paint the square in something as well as I do with my image, the server works too much, and does unnecessary work painting the same pixels of the image with a moved square, then it is not possible.

There is an interesting idea, what is wanted is a square, elements are always treated as squares. Javascript for better performance as well as a method to resize it are also wanted. It's just all done by WContainerWidget, the element used lots of times in the framework. As explained before, this element has a lot of advantages:

- It's already a square
- Can have a border and alpha channel for a pretty selection tool
- Uses Javascript
- Resize is a method already implemented.

There are also some problems.

- A WContainerWidget needs a parent. A possible solution to this problem has been to convert WebImage instead being a group of the two elements before mentioned, being a WcontainerWidget, with two children, and also the square selection. By doing this, we will have a clear parent that fits perfectly with every image.
- Controlling the borders: The selection tool should be controlled by taking account of the dimensions of the image, for not going outside it.
- Using CSS tools into Wt. The way to resize the right and bottom borders would be easy with the resize with Wt, but if what is wanted is to resize the left and top borders, should be used resize from Wt, and also move the square. The same for the borders, if a border is detected, the cursor ought to change to something clear to indicate that you can resize that border.
- Controlling all pixels behind the square. It's just a matter of programming, but it is something to take account of.
- Rotating the tool: This could be a difficult issue. There is a way to do it with CSS3 but not with earlier versions of CSS.

```
#footer{
margin-top: -28px;
-webkit-transform: skew(-33deg, 7deg) ;
-moz-transform: skew(-33deg, 7deg) ;
-o-transform: skew(-33deg, 7deg) ;
padding:15px;text-align:center;color:#dcdcdc;font-size:13px


}
```

This is a piece of CSS 3 code, which inclines the footer seven degrees. It could be useful to our selection tool, when using the incline function, every time it could rotate the div a degree, so it allows to be able to rotate the square. The main problem is that pixels behind the selection have to be controlled.

Selecting a region from a brush trace

It is another very interesting tool but there is no enough time to implement it yet. It is doing a selection by painting over the picture. All regions touch are just selected.

The difficult part is to make the trace, because remember that it's a image, it is not a canvas.

Extension to video processing

Streaming Techniques

Proprietary techniques

Adobe's Dynamic HTTP streaming

It uses an own protocol called Real Time Message Protocol (RTMP) to put the fragments into F4F files which are used, in addition to F4M which carries all the features of the files encoded at multiples bit rates to do a smooth streaming.

To use this system, it's necessary to install a plug-in in the client. Furthermore, there is a developer framework with a complete developer's guide.

Apple's HTTP live streaming

A division of the main stream is done in equal length segments and stored (.ts). Also is created an index which contain a play-list of all the media divisions and all the meta-data associated. The extension of this file is .M3U8

This index is served to the client and thus, it can control all the streaming process.

Microsoft's smooth streaming

In this method, chunks are contiguous in a mp4 file for random access. There are two different formats:

- .ismv: Contains both audio and video.

- .isma: Contains only audio

Server Manifest File (.ism) control the relation between all media tracks, bit rates and files, while **Client Manifest File** describes all the available streams of the client, the coders that can be used to be after decoded, video resolutions, etc.

Clients just request the fragments by the URI, using the quality and the offset to the next fragment.

On account of being a proprietary method, it also need to have the "media player" installed in the client.

Open techniques

Adaptive HTTP streaming (AHS)

It's a project from 3rd Generation Partnership Project (3GPP) which consists in divide the whole stream in chunks.

Usually to codify a frame, video coders use the previous frames, so in this case we should define all the group of pictures(GOP), starting from an I picture and ending before another one, as a chunk.
Of course, there is a meta-data file which indicates how to use this little parts of video, which are each one in one different URI. The name of this file is Media Presentation Description.(MPD)

When the client get the media presentation description, it generates other list, this is a segment list, which allow it to play the whole stream, divided in chunks in a fast way.
This version is the 9th release.
The standard document can be found at 3GPP web page in the section 26.234.

Dynamic Adaptive Streaming over HTTP (DASH)

The 10th release of 3GPP which is an improvement of AHS which includes new features as well as on-demand streaming video or linear TV including live media broadcast.

It consists, like the 9th release of a media presentation description which defines the streaming session.
MPD contains Periods, which are some time intervals. Inside each period, there are Representations, which are the different encoding ways the video stream has to be changed if some environmental issues, like bandwidth

decrease, are produced.

Inside this Representations we should find “segments”. There is in each representation, either a initialization segment and more than one media data segments or the media segments can be self-initializated if in the representation it's just one. For instance, the last segment of the video stream.

Open IPTV forum's HTTP adaptive streaming[32]

Open IPTV forum has adopted the 3GPP AHS and it's base to add some support to MPEG-2 transport files.

32 The Open IPTV forum's HTTP adaptive streaming standard

VLC(Videolan)[33]

Figure 37: VLC player from Videolan

VLC is a video player from Videolan which is actually able to stream audio and video using a lot of protocols such as TCP, UDP, RTP, HTTP, etc. Apart from playing all the media as a multimedia player.

VLC Streaming server is an easy way to do streaming trough remote machines or web pages.

It uses a library called live555 Streaming Media for all the streaming purposes. This library contains all the functionality needed for streaming to a web page.

The VLC source code is implemented in C, but it has also Java and Python bindings which means that you can write code in Java and Python if you prefer. There is a Javascript API too.

In the Videolan web page you can find all the information needed to getting started programming.

Gstreamer[34]

Gstreamer is a low-level library which allows you to create a lot of multimedia functionality in a plug-in structure. That is important because all the Gstreamer architecture is based on **elements.**

Figure 38: Gstreamer, a multimedia library

By connecting elements you can make whatever you want, for example play a OGG/vorbis file. Of course you need some elements like a OGG demuxer and a vorbis decoder and some other elements that I'm not going to

33 http://wiki.videolan.org/Developers_Corner
34 http://gstreamer.freedesktop.org/

mention at this point.
Gstreamer has very good utilities, I'm going to talk a bit about **gst-launch**: gst-launch is a very useful command-line utility which allows you to test your designed pipelines without programming a single line of code. Then, you can assure that your program will work without too much programming.

It's possible to create new plug-ins but the strong point of this technology is that each plug-in has to be independent from the other ones.
It also can do streaming which the reason why Gstreamer is in this point, but it has still some limitations such as HTTP streaming.
Flumotion is a open-project of software from the Catalan company *Fluendo* which allows you to do HTTP streaming, with a low-level API completely based in Gstreamer, written in Python.

Ffmpeg[35]

Figure 39: FFmpeg multimedia library

FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video. A lot of encoding and decoding methods are allowed, actually, the new release includes a lot of thinks to work with the new web technologies like HTML 5, as the name of this release indicates: “Works with HTML 5”.
The importance of this project to my final thesis probably is the possibility of different type of encoding methods including VP8 which is one of the most actual encoding systems whose Google is the owner, as well as the WebM container from the same owner.

35 http://www.ffmpeg.org/

Video demonstrations

Before going to image demonstrations I was finding out how to perform a live demonstration, how to do streaming.

In this part I was looking for a framework which had an easy way to stream. First I thought in VLC, but VLC finally become very difficult and I cannot find useful information for my goal.

Then, I've decided to change my mind, moving myself to Gstreamer.

Gstreamer is a framework with great documentation and also tutorials. It's really cool.

The thing is, I wanted to use HTML 5 video tag listening to one IP and port in the client. And In the server I wanted to have a program streaming the result of the video demonstration.

When you are working with video, you need to code and multiplex this video, because it's too big to be display as the user would like. Gstreamer has currently plug-ins to code in VP8, google's coding, and WebM, google's multiplexing way. So I wanted to choose between this and h.264 coding method.

But it's not important now, It's just that the better you code, the better you are going to see the video demonstration performance.

Actually there are more problems an is for this why I decided first just to do image demonstration, and in general data demonstration, but I'm going to tell you those.

Usually servers use port 80 for the http protocol. I mean, there is a firewall blocking connections on other ports. This is why we can't stream over other protocol, we just can stream in http.

With gstreamer, I've been able to do something like this:

gst-launch -v videotestsrc ! theoraenc ! oggmux ! tcpserversink host=192.168.1.35 port=8080

This is just a little program from Gstreamer to test your application before you code it. So I'm just sending TCP packets to the host which is going to see the video.

Then, I open the web browser in the client's host, and I go to my HTML page which has this HTML 5 video tag with the server address.

And it just works!

These are very good news, but unfortunately there is **no enough time** in this project to do video demonstrations although I'd have liked it.

Bibliography

Really to implement the whole project it has not needed a lot of bibliography. Wt API documentation is very good and it in addition to the forums where you can see all my questions that Wim and Koen(Wt developers) have answered me.

http://www.webtoolkit.eu/wt

http://redmine.webtoolkit.eu/wiki/wt

http://www.webtoolkit.eu/wt/doc/reference/html/index.html

This is the wiki of the UPC image and video processing group, also useful as well as the ImagePlus API documentation.

https://imatge.upc.edu/wiki/

https://imatge.upc.edu/wiki/pmwiki.php?n=ImagePlus.APIDocumentation

About the rest of the pages I've looked for to do the project, all them are in the footer of the page in which explains the concept, because of an easy understanding of the reader.

Annex

Documenting getting started

Here there is a piece of code to illustrate how the getting started was made.

It is indeed, a little tutorial.

```
\section WebApp Creating a web application


This section is just to give you a brief introduction to the web
framework. The main goal of this framework is that with a few
lines of code,

you can achieve a complete web demonstration within a predefined
web interface, including in it the header, footer, images etc.


Well, let's get our hands dirty.

This is the code to have your web application in an hpp file.
It's a new class that inherits from GPIapp, which is a web
application.
```

\code

```
#include <imageplus/toolbox/web/GPIapp.hpp>

class NewApplication : public GPIapp
{
    private:
                //Here is where you should put your new objects.
    public:

        //Constructor
        NewApplication(const WEnvironment& env);

          //Here is where you should put your new methods

};
```

\endcode

By doing this you will have a whole application, and what is missing is, your web demonstration inside it!

But we also need a cpp with the main, which is always the same in all web applications.

\code

```
#include <Wt/WApplication>
#include <NewApplication.hpp>

WApplication *createApplication(const WEnvironment& env)
{
  return new NewApplication(env);
}

int main(int argc, char **argv)
{
  return WRun(argc, argv, &createApplication);
}
\endcode
```

Another important thing is missing, the title and description of the web application are MANDATORY, so we should put them in the constructor like this:

```
\code

NewApplication::NewApplication(const WEnvironment& env) :
GPIapp(env,

"New Title of the Web Application",
"New piece of text, as description for the application")

{
     //Your code
```

```
}
\endcode

\section Layouts Working with layouts

Instead using CSS which is another technology to learn, we thought it would be better to use layouts because the style is the same than in Qt.

You can see here <a href="http://www.webtoolkit.eu/wt/doc/reference/html/classWt_1_1WBoxLayout.html">Layout Documentation</a>

There is indeed a thing to highlight, the layouts have to be attached to somewhere. Your web class has itself a method which is getDemo() that gives you the main parent for your "layout master".
Then you can put whatever you want inside. Here there is a little example. (In the constructor)

\code

//Create the layout which is defined before in the class definition.
layout =  new WHBoxLayout();

//Create the text which is also defined.
_text = new WText("Hello I'm a text");
```

```
//Add the widget to the layout
layout->addWidget(_text);

//Attaching the layout to the WebApplication.
getDemo()->setLayout(layout);

\endcode

\section Programming About the programming style

There is a clear conflict between the programming way of Wt and the object oriented programming. They actually work with pointers.
And the web demonstrations should be done with pointers to avoid double-free problems.
Here there is an example:

\code

WText* plain_text = new WText("Hello");
root()->addWidget(plain_text); // root() is pointer to the parent container of the demo
                               // you should create (sub) containers and layouts
\endcode
```

Here you are just using pointers, finally you get a pointer which points to a WText element.

The important thing here is that you never have to "delete" anything because the Wt containers do it for you (if you have added it by addWidget).

\section Wt Useful things from Wt

- WText: A simple text Text
- WPushButton: A button A Button
- WContainerWidget: This is the 'div' container in html. An HTML division or container
- WComboBox: This is a box where you can have many choices. Combo box
- WRadioButton: A radio button. Radio button

\section Running How to run a web application

Here there is an example:

~/workspace/imageplus/applications/web/NewApplication/bin/release/NewApplication --docroot /usr/local/etc/wt --http-address 0.0.0.0 --http-port 8080

Well, let's divide it in parts.

At first, our executable

~/workspace/imageplus/applications/web/NewApplication/bin/release/NewApplication

The root of all the resources, it's shared, so it's always the same.

--docroot /usr/local/etc/wt

The HTTP address is the IP address where you want to run the application

--http-address 0.0.0.0

The HTTP port is the port where you want to run the application

--http-port 8080

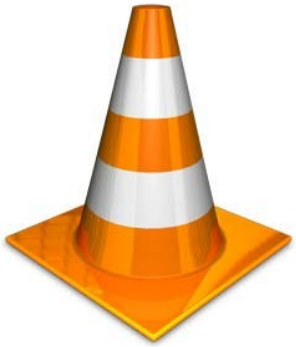
After that, it will report how to enter in the application. But well, it's not difficult, just go to the web browser and put your IP:port and the application will be running there.

In this case and as an example I should open http://0.0.0.0:8080

Bitsearch blog posts

Friday, September 18, 2009

How to download and compile VLC source code



I, like Albert, Pia, and Christian(Who is writing in the blog too) have started to do some projects with the intention to participate in the next Google Summer Of Code edition.
We are studying "bachelor in Electrical Engineering focus in multimedia" and for this reason, my project is about VLC from Videolan, one of the projects that google offers in their GSOC page.
I have to understand how it works, for programming some aplications.

My goal is to do a miniproject from Videolan. But, VLC's source code is changing constanly, because there are a lot of programmers writing source in parallel. I've worked with my Ubuntu 9.04 Jaunty Jackalope. For solve this problem, In Linux there is a tool called "git" which allows to get the last version of the code in that moment. I'm going to write all commands that i've put in linux's command line,at first to compile VLC, and then to install git. Git is a tool to control that you are working with the last version of the source code, beucause if I starts to write code without any care, when I compile, It can be that source code has changed a lot, and because of that, my new contributions could be wrong. Here, you can see the official webpage of git. I've compiled VLC with this commands:

```
wget http://download.videolan.org/pub/videol ... .
6b.tar.gz
```

Then, i've moved vlc to my desktop.

```
cd Desktop
tar xvfz vlc-0.8.6b.tar.gz
cd vlc-0.8.6.tar.gz
```

```
sudo apt-get build-dep vlc
```

then VLC is compiled,now, i'm going to install git:

```
sudo apt-get
git
```

Then once git is installed, you need to clone the vlc repository.

```
git clone
git://git.videolan.org/vlc.git
cd vlc
./bootstrap
./configure
make
sudo make install
```

When I tried to execute ./bootstrap I had some problems with the packages. There were no all the packages which vlc needs in Ubuntu.I looked for in my synaptic and I found them.

VLC and git are already, ready to work. The last point is about every time that I want to start writing code, I have to get the last version of VC's source code. This is so easy like writing in the command line:

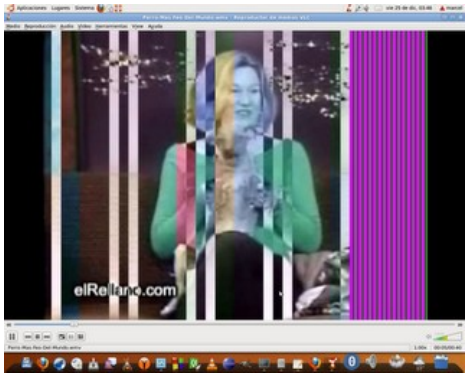
```
git
pull
```

When this aplications are compiled, my next step is to look for a miniproject in the VLC's page. I've seen a miniproject about video filters that I think is very interesting.

I will inform you! Regards!

Video Filters in VLC

In VLC, there are a lot of filters implemented and you can access them by the GUI,
the graphical interface.
The path is: Tools -> Effects and filters --> Video effects
The other way to access to this filters, by the command line is this:



vlc --video-filter name_of_the_filter

When you try to start any filter with this command, a new instance of VLC starts, but with the filter you have chosen charged.

For example: If I want to apply the filter "invert.c" the command will be:

vlc --video-filter invert.c

After this introduction, I've started to play with the filter "invert.c", which is written in C, trying to understand it.
I've achieved to invert some of the pixels in the image, by choosing which pixel i want to change.

I'll explain a bit how VLC works with the image when have to apply a video filter.
The way that vlc paints the images is grouping them each 64 pixels. Each line is composed by a lot of this groups of 64 pixels, and in the end, by a part in which vlc paints pixel to pixel.

My intention have been try to invert the odd numbers and try to paint the part which is painted pixel to pixel with a different color to see how vlc uses the values of this pixels.
However, I have tried to paint some different pixels, in the pixel to pixel zone, and in the main zone.

After that, I have had to compile vlc again. I've compiled it with the commands:

cd ~/vlc make sudo make install

This commands compile the full vlc code, but I'm searching the way of compiling only the filter "invert.c"

Streaming from VLC to VLC in the same computer

My goal at this moment is to create a C++ program which streams a video using libraries (perhaps VLC libraries, it is not defined yet), and finally displays it in the web, using HTML5.

To do that, as a first step, what I've tried is to stream from my computer with the "local loopback" whose IP adress is 127.0.0.1 to the same computer. This step has been because probably if I stream computer to computer I'll find a firewall blocking my stream.

The whole process has been:

As server:

1.Open vlc
2. Click media -> Streaming
3.Select the file and click add
4.Click stream
In the following window, click next.
5- I've checked "Display Localy". This is for if you want to see what you are streaming. But it makes the streaming slower.
6-select http and click in "add"
7-Write the local loopback ip 127.0.0.1
8-The port, 8080.
9->Activate transcoding to Vorbis+Theora(OGG)
Then click next, and click stream

Your vlc must be streaming now.

As client:

1.Open another vlc.
2.Click media -> Open Network stream
3.select http and put the local loopback ip 127.0.0.1:8080

Now you'll see your video streaming.

The input file was a .avi file but I think it doesn't matter because we've checked the box "transcodificate" and the video stream will be .ogv with Vorbis as audio and Theora as video.

Thursday, November 4, 2010

Streaming with VLC directly to the web

My intention was to stream with VLC. Then I achieved to receive this stream with another VLC. Then, the next step has been to display this video stream in the web using HTML5.
Not so much browsers support html5. So I've chosen Google Chrome and it has been possible to display the video stream.

I've streamed the video through http as I said in my last article.

This is the HTML5 code that I've used:

```
<video src="127.0.0.1:8080">This browser doesn't support HTML</video>
```

Another important thing has been de DOCTYPE definition, because in HTML5 the DOCTYPE is different from HTML Transitional.
The HTML Transitional DOCTYPE is this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

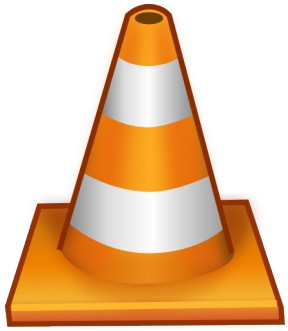
Then, the HTML5 doctype is

```
<!DOCTYPE html >
```

So much shorter than the HTML Transitional.

The next step will be to investigate about the way how VLC streams. The library it uses, and alternatives.

Introduction to the Gstreamer System Architecture



In my last post I wrote some different things about VLC, which I wanted to use for streaming purposes. I had to investigate what library it used. Actually, there is one library called "LIVE 555 streaming media" which the cone, VLC, is using.
Really, it was very difficult for me to find information for getting started.

For this reason, it was proposed to me, to investigate a bit about another low-level library called Gstreamer and written in C, for streaming and playing multimedia, which is used in several applications.

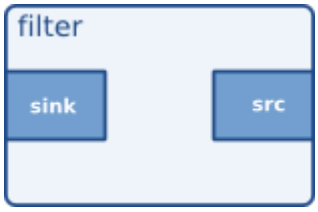
The first impression was really good. There is a first manual in which you can learn the basic features of Gstreamer, how to initialize it, how to put it in different state, what is a pipeline, what is an element, and finally a OGG player, as a "Hello World" application to apply the concepts already learnt.

As a result of my investigation, I'm going to summarize what I've learnt in this Gstreamer Manual.

The main use of Gstreamer is to manage the video and audio stream to create, for example, a video player, or a streaming server.

An **element** is the most important class of objects in Gstreamer. Usually, a functionality is achieved by linking elements.

Normally, inside the elements, there are **pads**, which are inputs and outputs. The input pad is called "**sink**" and is from where data flows trough the element, for example, from a local file, or from another element. The output pad, is called "**source**".



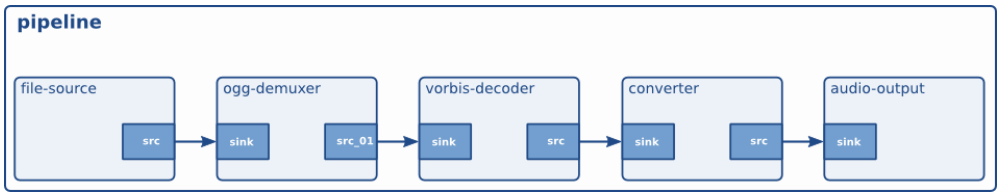
Here we can see an example of an elements with two pads. An input pad, and an output pad.This is an element called *filter*, with input and output pads.

Besides elements, there is a kind of container called **bin**, which would be the parent of some elements. And then, there is a type of bin, called **pipeline**.
This is a hierarchical structure for one important reason. A medium-sized project with gstreamer will contain a lot of elements, imagine that you want to put the "play" state. You should tell to each element. This method allows you to tell the command to the pipeline and it will advise all its elements.

There are also buses and message reporting functionalities which I'm not going to explain in this post because it will be too long.

Finally, as an exemple of the above explained, we have the first "hello world" application, which is an audio player of the Vorbis encoded content in

an OGG file.



In the next posts I'm going to explain how it functions by putting snippets of code, and extending a little bit more the bus and message functionality.

Open source software for video streaming on the web

In this blog post I would like to explain what is my final thesis about, and three possible actual technologies to succeed in.
It's about how to demonstrate the algorithms of the image processing group in a web interface, with the most recent technologies such as HTML 5.

To stream the video to a web page, we need a streaming engine. This, have to be able to stream the video, so we have to be aware about wheather the most common browsers support all this technologies.

I think that with a little figure is easier to understand:

Where we can see the source as the output of some video processing.

It's clear that all browsers accept HTTP protocol because all pages are served of this way.
And other advantage that HTTP protocol has between other protocols like UDP or RTP, apart from the support for technologies, is that firewalls usually don't accept this type of traffic.

Focusing in the streaming engine, there are three possible technologies to cover this area:

Gstreamer:

Gstreamer is an open source low-level library for multimedia streaming porpouses which is programmed in C. It's a very good designed library with a plug-ins structure which allows you to use a lot of functionalities.There is also an example of software called "Flumotion" from the company “Fluendo” for HTTP streaming. This software could fulfill this requirements.

Fluendo has one of its headquarters here, in Barcelona.

A few days ago I wrote an article explaining the basics of gstreamer.

FFmpeg:

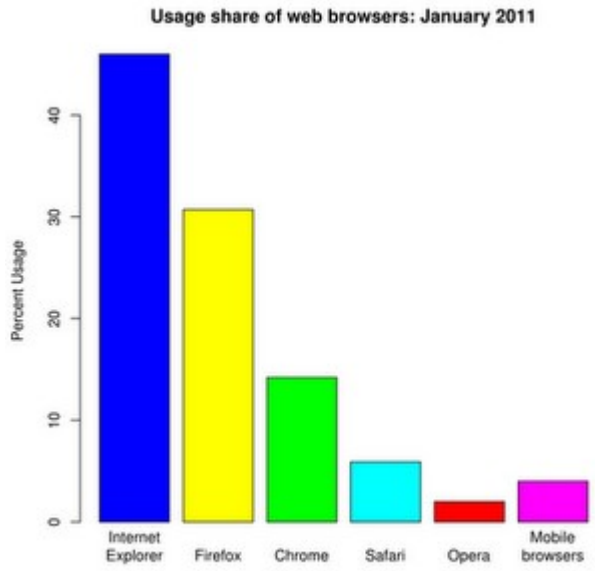
FFmpeg is also a complete solution for the manipulation of audiovisual content. This includes a lot of functionalities such as encoding, decoding, recording, audio and video streaming, etc.
Some of Gstreamer plug-ins are written with ffmpeg functionalities, using them in some of their plug-ins. In terms of my project, FFmpeg is not so much recommended because there are too much changes between different versions of this library. It could be the weakest point of this technology.

VLC media player

VLC media player is an open-source project for playing audiovisual content. It also has a streaming server which could be useful to stream to the web. VLC is programmed in C/C++ and the most interesting library for us is liveMedia, which is a set of C++ libraries and frameworks for multimedia streaming (RTP/RTCP, RTSP, SIP) . It has also some functionalities taken of ffmpeg.

We also have to be conscious about the compatibility of the browsers, the most important ones, at least.

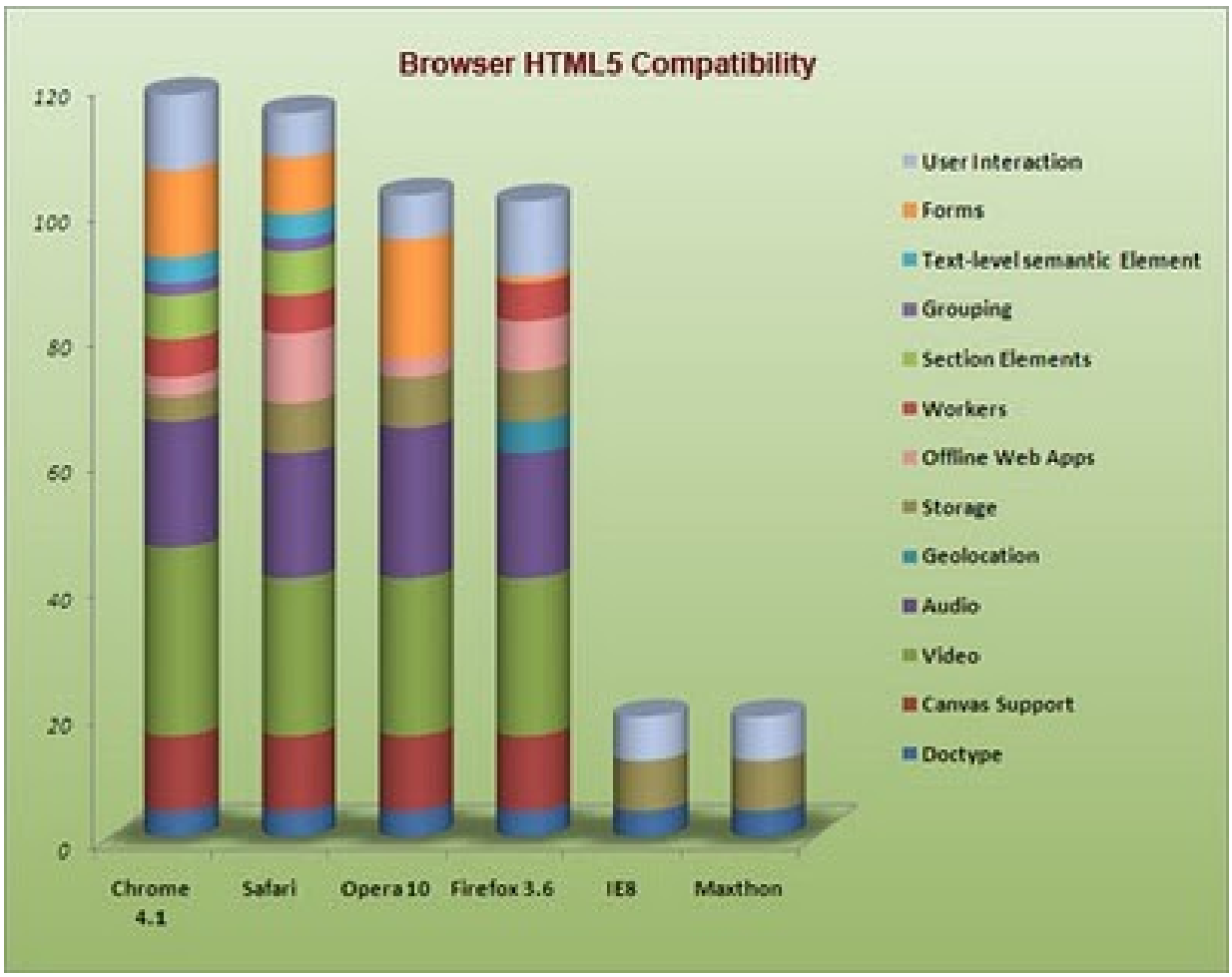
Here there is a comparative of the usage of browsers during January of 2011.



But the use of HTML 5 is not still implemented in all the browsers, in some of them.

Here, there is another figure with the HTML 5 functionalities implemented in the most updated version of each browser. My college Christian wrote an article about HTML 5 explaining what is and what news it includes. We have to look at the green one, which is support for the "video" tag which is able

to handle a video stream into a web page. As you can see, In Chrome 4.1, Safari, Opera 10 and Firefox 3.6 this functionality is already implemented.


Browser HTML5 Compatibility
120
100
80
60
40
20
0
Chrome 4.1
Safari
Opera 10
Firefox 3.6
IE8
Maxthon
User Interaction
Forms
Text-level semantic Element
Grouping
Section Elements
Workers
Offline Web Apps
Storage
Geolocation
Audio
Video
Canvas Support
Doctype

Adaptive Streaming over HTTP

How do we can watch a video on internet? There are a few ways to do that. We can just see it by **downloading the whole video** file and after, playing it.That's not comfortable for the user experience's because usually video sizes are not small, and depending of the bandwidth of the connection it can take a lot of time.
There is another way called **progressive download**, it consists in a service in which you can download the video and play it at the same time. In this method, the video has to be stored in a server, it cannot be streamed by a live web cam.

There is a lot of discussion about what streaming is, and what is not.

The definition of streaming says that just the needed parts of media are transferred in order to play them. I mean, in progressive download, services like *youtube*, when you pause the video to do other things, this video is still downloading and wasting valuable bandwidth

In addition, returning to the title, it says adaptive. Why adaptive?
Imagine that you are doing streaming with a web cam. Suddenly, the net becomes more and more used, and your video is just being played slower because of the limitation of the net's bandwidth.
Adaptive then, means that the scalability of the video can change depending on the bandwidth available and affecting the spatial (pixels per unit of width), temporal (frames per second), quality (DCT Coefficients and module of Motion Vectors), or combinations of them.
Of this way, in the case you were watching a video and the bandwidth were variable, there would be smoother than before.

The other interesting issue is **why to use HTTP**. HTTP is the protocol used to web pages, but there are some reasons that make it suitable for sending video through it.

- Easy and effortless streaming by avoiding NAT and firewall issues.

- There is a clear trend in the web to send video trough HTTP. All Apple's devices, and mobile devices general, you tube and more web streaming services.
- It has the ability to give the control of the streaming to the client.

Seeing that, despite there are no standards yet, some companies and entities have defined some ways to do streaming over HTTP. Here in the Bitsearch, you can see another post with some of the most important technologies.

Thursday, February 24, 2011

HTTP Adaptive Streaming Techniques

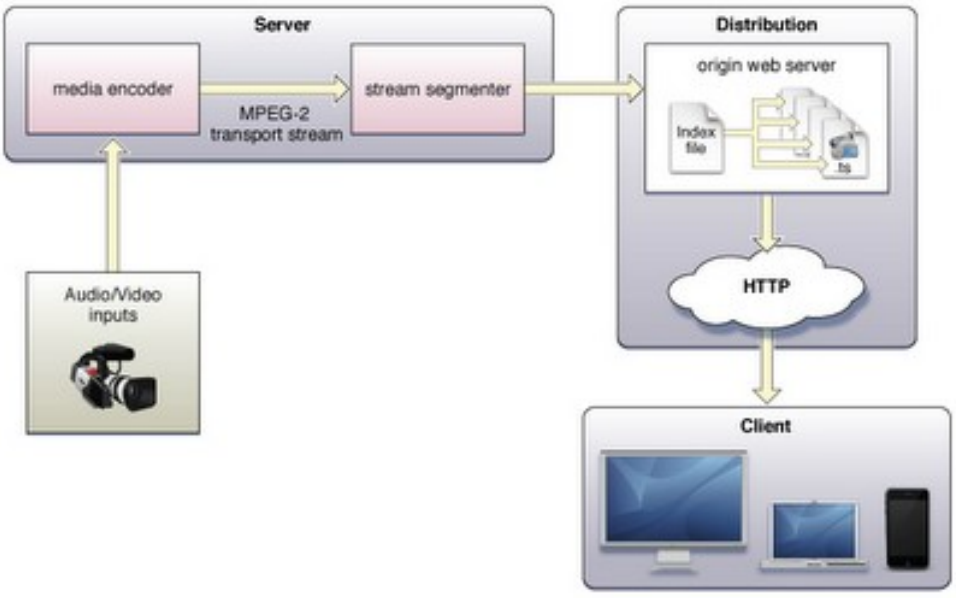
There is no a clear and standard way to do streaming, but there are a few techniques to do it. At this point I'm going to summarize them.

Proprietary Techniques:

Adobe's Dynamic HTTP streaming

It uses an own protocol called Real Time Message Protocol (RTMP) to put the fragments into F4F files which are used, in addition to F4M which carries all the features of the files encoded at multiples bit rates to do a smooth streaming.

Apple's HTTP live streaming



A division of the main stream is done in equal length segments and stored (.ts). Also is created an index which contain a play-list of all the media divisions and all the meta-data associated. The extension of this file is .M3U8 This index is served to the client and thus, it can control all the streaming process.

There is a final thesis of Bruna, another student of my career, which explains perfectly all the apple's HTTP live streaming for more information. The document is written in Catalan.

Microsoft's smooth streaming

This method's chunks are contiguous in a mp4 file for random access. There are two different formats:

- .ismv: Contains both audio and video.
- .isma: Contains olny audio

Server Manifest File (.ism) controls the relation between all media tracks, bit rates and files, while **Client Manifest File** describes all the available streams of the client, the coders that can be used to be after decoded, video resolutions, etc.
Clients just request the fragments by the URI, using the quality and the offset to the next fragment.

Open Techniques:

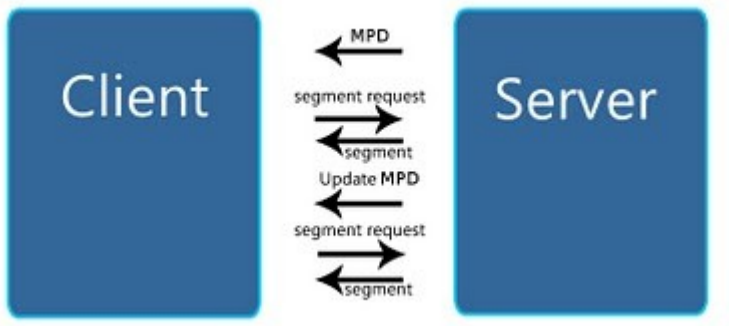
Adaptive HTTP streaming (AHS)

It's a project from 3rd Generation Partnership Project (3GPP) which consists in divide the whole stream in chunks.

Usually to codify a frame, video coders use the previous frames, so in this case we should define all the group of pictures(GOP), starting from an I picture and ending before another one, as a chunk.

Of course, there is a meta-data file which indicates how to use this little parts of video, which are each one in one different URI. The name of this file is Media Presentation Description.(MPD)

This version is the 9th release. You can see the whole standard in the official web page chapter 26.234



Dynamic Adaptive Streaming over HTTP (DASH)

The 10th release of 3GPP which is an improvement of AHS which includes new features as well as on-demand streaming video or linear TV including live media broadcast.

Open IPTV forum's HTTP adaptive streaming

Open IPTV forum has adopted the 3GPP AHS and it's base to add some support to MPEG-2 transport files.

Tuesday, March 1, 2011

Going through the HTTP protocol

When you are looking up a website, a lot of things happen. One of them is related with the HTTP protocol.

Let's talk about HTTP. Fist of all, HTTP is a protocol created by World Wide Web consortium and Internet Engineering task in order to allow all the elements which allow a web architecture to communicate themselves.

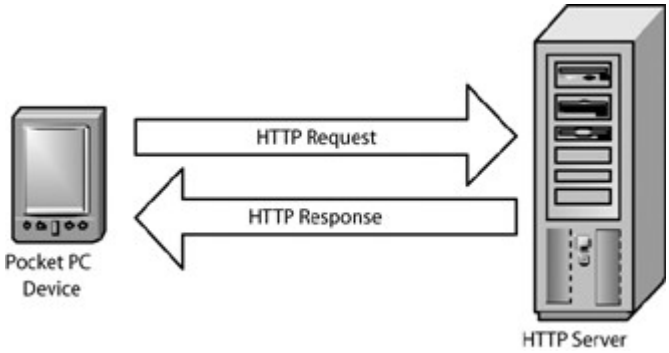
And then, how does it work?

Certainly, is not a very difficult protocol, it's based on request-reply scheme. This means that in a communication, first, the client will send a request, then the server will reply this fulfilling as well as possible.

It's important to highlight that it's a stateless protocol, because it doesn't keep any state, this limitation was the main reason for the introduction of cookies in web browsers, to keep a state in the client.

In the request, the client normally asks for some information, and the server gives it back. When you are in the situation above mentioned, looking up for a web page, the server has returned the whole web page in the reply.

But HTTP is not only to transmit web pages. It can be used for transmitting some attributes and values which can launch a CGI script in the server, for example.



Furthermore, and this is the most important part for my project, HTTP is used also to send video, avoiding the firewalls. Therefore, we could send video packets through the web, and take it with a HTML 5 interface.

But well, focusing in the practice, I'm going to show a real example of HTTP communication.

Once a connection has been opened at the server's URI and port, (i.e. URI: http://www.example.com/index.html, port:80) the client sends this message to the server.

```
GET /index.html HTTP/1.1

Host: www.example.com

User-Agent: name-client

[Blank line]
```

Then, the server replies with:

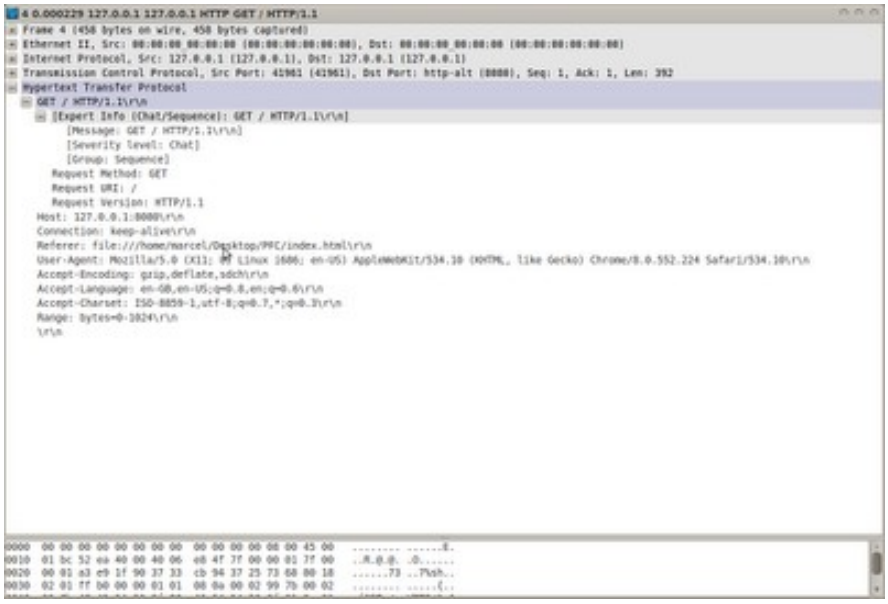
```
HTTP/1.1 200 OK

Fate: Fri, 31 Dec 2003 23:59:59 GMT

Content-Type: text/html

Content-Length: 1221

[...CONTENT...]
```

As you can see, the server is answering that the server's response has been right, with HTTP version 1.1, with the “OK” message, and the 200 code, which is a status code. Here is the list of status codes.

Moreover, the data is in the reply, always in GMT format, and also the content type, which refers to what mime type the data is, and finally, the size of this set of information.
The HTTP packet can have some attributes that can make the protocol work in some different ways. For example, there is an attribute, “content-encoding”, which explain how the data are encoded, or an authentication via HTTP.

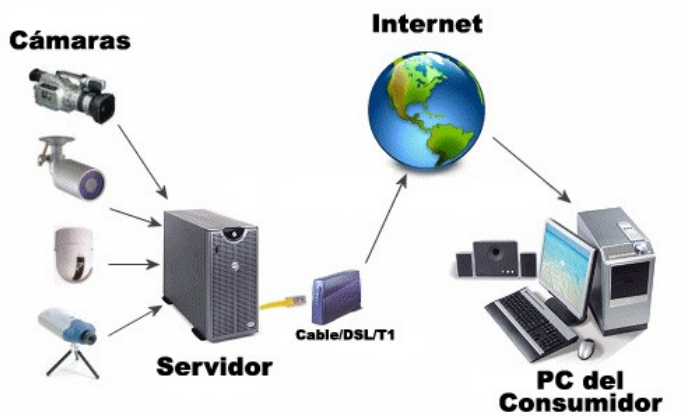
Here you can see the whole list of attributes that client and server can request or reply respectively.

As an example, I'm going to show you HTTP headers of a traffic capture that I've done with Wireshark while I was streaming a video with VLC(Videolan) and capturing it with a HTML 5 interface.



Wednesday, March 9, 2011

Streaming to an HTML5 web page



In some articles before this, I've been talking about some separated issues, such as Gstreamer, VLC, the HTTP protocol, streaming techniques and HTML5.

Now, I've achieved in streaming against a web page. And I'm going to explain how the whole test has been.

First of all, I succeeded in sending an HTTP stream with VLC, which has been a very important tool for my last discovery because it has allowed me to investigate how it works.

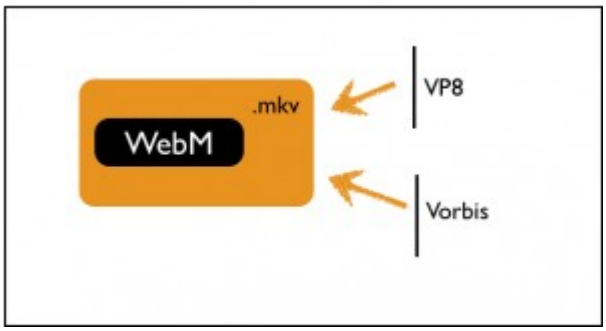
Another important tool to take account of is wireshark. Wireshark is a network software that allows you to look at and capture what is going through your network interfaces. So you can see all the packets and a lot of information about them, for instance, how the protocol stack is composed.

Besides, I have to reference you again to another article written by myself which explains the main architecture of Gstreamer, one of the most important frameworks for streaming purposes in this moment.

Once said this, let's get our hands dirty, let's to streaming!
Firstly, we have to think about who is going to do streaming, and how. Now we are assuming that we are in the server, and we have to build something like a Gstreamer pipeline.
This pipeline has to cover some requirements.

It has to start with a well-known video file and format. It will be raw.
We have to be aware about what encoding and multiplexing methods HTML5 supports.
What does HTTP streaming means in terms of the HTML5 video tag.
Well, step by step. The first part is what to send. It is an early test so, we can send a gstreamer video pattern already designed for test purposes. It's name is "**videotestsrc"**. We have got the first piece of the puzzle.

Now, our current worries are the the encoding and multiplexing methods. Well, really, when I tested the VLC HTTP streaming a few time ago, I used Theora as video encoding method,vorbis as audio encoding method, and ogg as well, as multiplexing multimedia container. It worked perfectly, therefore I'm going to use them to do this test. All them except vorbis because there is no audio in here.



Then, I'm going to use theoraenc, the theora encoder, and oggmux, the ogg multiplexer. Of course in the future theora can be replaced by h.264 or VP8 and ogg can be replaced by other containers like webM or mkv.

And finally, the last and for me the most difficult part of this test, to send this coded and multiplexed video that we have got, through the red.
Well, as you know internet is an end-to-end global net. It means that you can't broadcast like if It were a TV broadcasting. Ok, at this point, I did some tests in the way of how could send this data. I spent a lot of days investigating the HTTP protocol and how I could send this data through this protocol, like VLC does.
The strange thing was that when I was capturing the traffic produced by the

VLC streaming server, there was no traffic.

Then, in the other machine, opened my HTML 5 file, and then, suddenly a lot of HTTP and TCP traffic were produced.
When you looked at the traffic, you could see the TCP handshake, and then a get request from the client to the server.
It made me think that perhaps the video HTML5 tag manages to encapsulate the data in the HTTP protocol in order to be displayed in the web page, and we just have to send the data via TCP. And effectively, this worked.

To sum up, in my ubuntu, I executed:

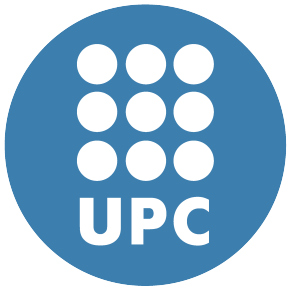
```
gst-launch -v videotestsrc ! theoraenc ! oggmux ! tcpserversink
host=192.168.1.35 port=8080
```

If you have read the blog posts before, you'll know what is gst-launch, the -v is to verbose all the information, the ! sign is used to indicate that this elements are connected, and the las element is a TCP sender, which sends to this IP and to the 8080 port. This is running in the server's machine.

In the client machine I just have to open the HTML5 web page with a browser like Google Chrome 8 or Firefox 3.6 which accept the HTML5 video tag.
If you want to be sure that is being correctly sent, I recommend you to check the traffic in both points of the connection, in the client and server.

Right now, if all has been right, you ought to be doing streaming to a web page using Gstreamer and HTML5

Wt - Web programming with C++

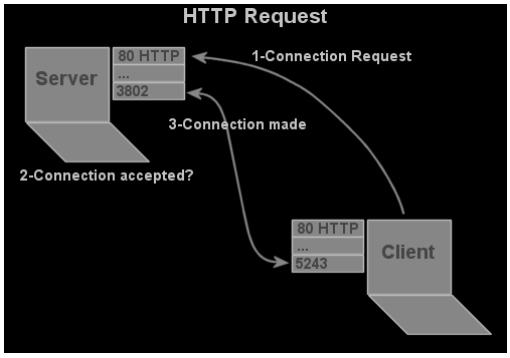


Some time ago I was thinking in how to do a demonstration of a video algorithm for the UPC image processing group. The first idea was to display the video output in a web interface. I certainly thought a lot around this issue, as well as investigated technologies such gstreamer, ffmpeg, etc.

But really this is not the only way to demonstrate those algorithms. Being aware that not all algorithms output a video, that some of them output coordinates or data in general, it's easy to think that there is another way. What about data streaming.
On the one hand, there is the advantage that we can forget for some time the encoding and multiplexing processes.
On the other hand it's still streaming, so I ought to find out the way to do that, with a smaller amount of data in each sent.

With this last idea, my project became a bit different, the data now are not frames, the data are whatever, coordinates, names, etc. For this reason, I had to research how to send these data.
In a image processing algorithm, is not known when data are going to change, so we certainly need a requirement, data have to be pushed.
I mean, imagine the server in one side, the client in the other side.
Then the server **just gives data** to the client whenever it wants, and the client just collect this data and integrate it in the web page live demonstration.

Well, I think this is very easy to understand but not possible theoretically, and I'm going to explain why not. We are trying to send the data trough HTTP protocol. This is because HTTP protocol is allowed by firewalls and we can send data without being blocked.



Then we have to think how HTTP works. The way how it works is sending a request to the server, typically a GET or POST request, and the server responds with the data.
So, as you can see, negotiation is needed, we can't just push data. It will be incoherent.

Nevertheless, there are some hacks, you can have a look at reverse AJAX, which has a lot of names, such as comet programming, or a lot more.

I'm not going to comment all of them, but I'm going to mention the most important one, long polling.

This hack consists in doing a request (the client), and **when the server has new data**, then respond this request. When data arrive, a new request is sent. Thus, we can have a more efficient way that steadily polling.

Actually. there is a library called Wt (uttered as witty), whose use is exclusively to web programming, with the only difference that it is a C++ library. This library is very similar to the famous Qt C++ library which is used to create desktop interfaces.
The library has a very good documentation, very good forums, and in general the impression that it give as, me, Albert and Xavi was a great impression. We couldn't believe that this library was able to do all what says in the documentation. It was really strange, if so, why is not it very known?

Our final answer was that is not very known because it is in C++.
Nowadays, web programming is made by some technologies like PHP, Javascript, AJAX, ASP.NET, and so on.
But Wt offers the possibility to C++ programmers to make a web page without any knowledge about all of this technology. Abstracting thus, these technologies just by using C++.

Well, once said this, I'm going to mention my current trial applications.
The first one has been a simply text input that when you key up, a message such “Hello unknown, how are you?” pops written down in the screen. When the name is known, it shows this name instead unknown.
The background of this simple application is that every time you key up, there is a request. In addition, I've managed to learn how the style works in this library.
Here I leave a screen-shot of my little application:

This has been a good starting point, to realize how Wt could render a web

page using C++. But after doing it, I've gotten my hands dirty doing a more complex server push, based on the example server push. Finally I've included another point to my second application. This is the painting.

The second experiment has been to demonstrate the random algorithm from stdlib of C, by sending two random data, in a random time interval, and then the data are rendered by painting a little circle in the coordinates (random1,random2).

I think that it is a good approach because I've already achieved to push data through the net. Now, it's time to connect my little application with ImagePlus, and take over the C random by the ImagePlus random. This will be my first ImagePlus algorithm demonstration.
I expect to show you this in my next post, so, have a look!!

Thursday, March 24, 2011

My first Wt C++ web page

The main goal of my project, is to show algorithms in a web page. If I want to do so, I obviously need to create a web page. But, what is the best way? There are a lot of technology behind web pages and web applications, PHP, ASP.NET, Python, ruby, etc.

In my last post I spoke about Wt, a C++ library for web purposes. In addition, ImagePlus, which is also programmed in C++, is the library used by the Image and Video Processing Group from the Polytechnic University of Catalonia.

What I'm doing is trying to make them easier to show their algorithms in a web page. Doing it in other language different from C++ would implicate someone would have to learn web based languages.

Having said this, what I'm going to explain is that there is an important abstraction process between Wt and web technology. Wt programmers don't care about neither PHP nor any other web issues. So you can program whatever you want just using C++, which is very good for all C/C++ experienced programmers because the need of learning a new language for the web disappears.

And, what about HTML?
I've been programming some time with PHP, and like it, what you are doing is writing the HTML code to be displayed in the client. But in different ways. In Wt you can create a new element such as a WContainerWidget, and it's supposed that you have written a "div" element into a HTML web page. But the difference is that WContainerWidget is a class and you can use its

methods in a very easy way.
Conversely, in PHP you have to print all the page code as an string, so it's not so fine.

And CSS, How does Wt manage to put the style on?
First of all, CSS means Cascading Style Sheets and its purpose is to split content from style. Wt allows to programmers to apply the style in two different possibilities. You can apply it in some methods directly from C++, or you can add an external CSS style-sheet with all the rules.
Certainly, CSS is not a programming language, is most a descriptive language about how is the style of the page. Consequently, its syntax is very easy and it's not a problem.

Well, and right now, I'm going to comment a bit my first web page example. I've done five classes which are:

- GPIapp
- GPIpage
- GPIheader
- GPIcontent
- GPIfooter

All these classes except GPIapp are derived from WContainerWidget. In other words, these are “div” elements. GPIapp is used to launch the web page and set all the external style-sheets and features.

GPIpage is the main container. Inside it, there are the rest of the classes. The GPIpage class is composed then of four private attributes, which are: the page itself, the header, the content and the footer.
In the header we have the logo, a title, and a space in the right.

The content is the perfect holder where the demos are going to be placed. So probably, it will have to become a bit more complex. Another point of abstraction, I think that it will be good stuff if there were a class called GPIdemos, and the content could accept one of those, then the complex code perhaps will be in this last class. Well, this another step, but it has to be thought correctly, so, let me think and I'll explain you it in next posts.

This is my first web page example:

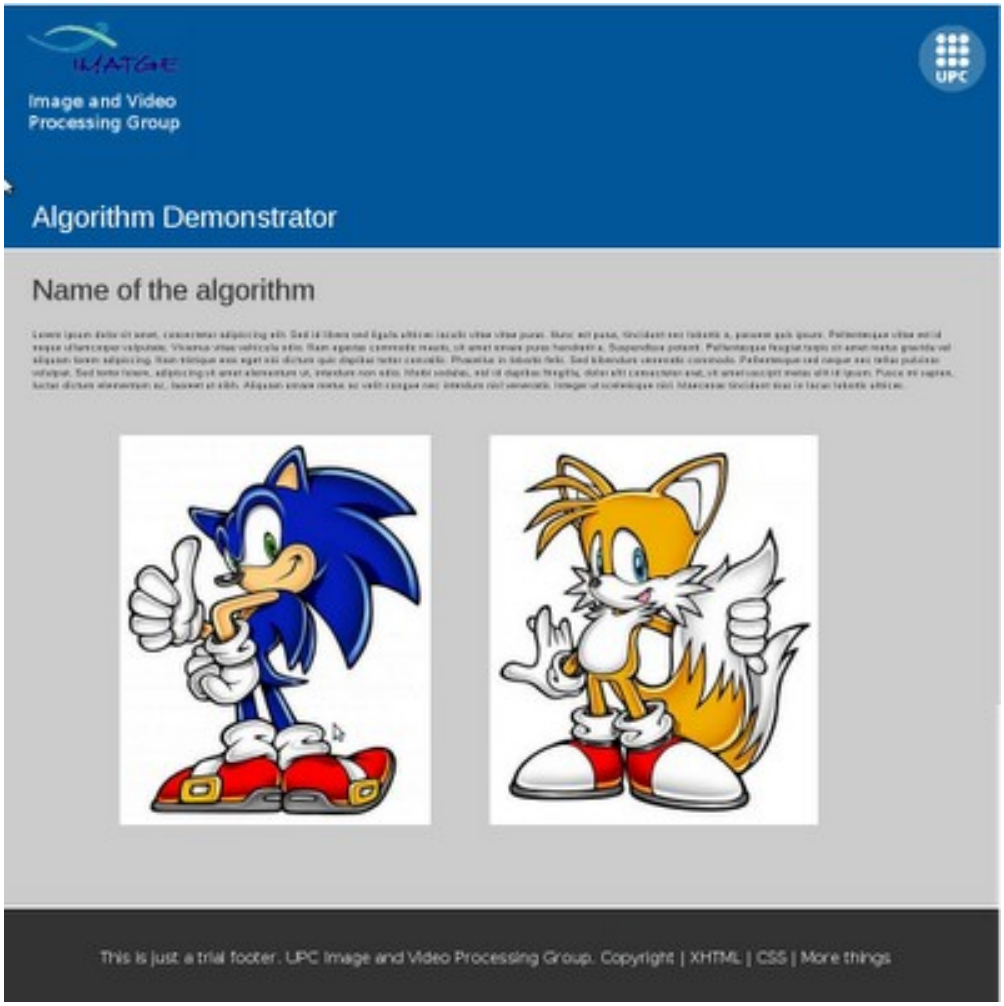


Image and Video
Processing Group
UPC
Algorithm Demonstrator
Name of the algorithm
This is just a trial footer. UPC Image and Video Processing Group. Copyright | XHTML | CSS | More things

Integrating Wt + Imageplus

These last weeks I've been getting started with a C++ library whose purpose is to build web applications. This library is called Wt, you can have a look at my last articles to know more about it.

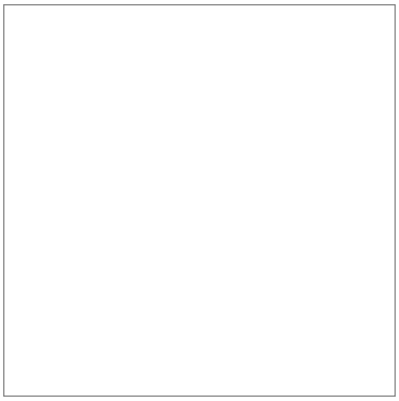
My project consists on demonstrating algorithms from the UPC video and image processing group, so an important step in the development of my application would be to be able to use imageplus algorithms, the library of the group, and create Wt classes in order to show them in a web page.

I have to thanks Albert Gil for helping me with the libraries, because the first step was to change the way of compiling the project, in Wt, actually they are using a program called Cmake, as well as make.

Imageplus, is currently using a program called scons to compile.

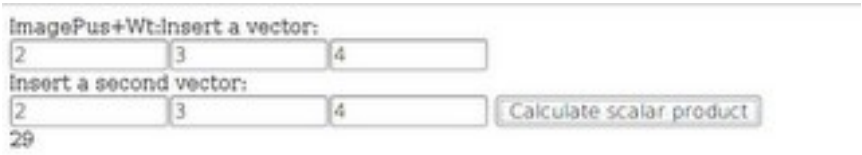
We changed this and landed our workspace to the IDE that they are using, Eclipse.

Furthermore, we had to add two libraries from Wt in order to join them, the name of the libraries is *libwt* and *libwthttp*.



Once environment was ready, I started to work, and I did some examples that I'm going to explain right now:

The first one was an easy one, we just wanted to test if it was possible to request some data, process it with imageplus and in the end, show the result in the client's browser.
And the algorithm I used was the scalar product between two vectors, which is a data input and data output algorithm, no images, no videos.
Well, you can see in this figure the result, it worked fine.

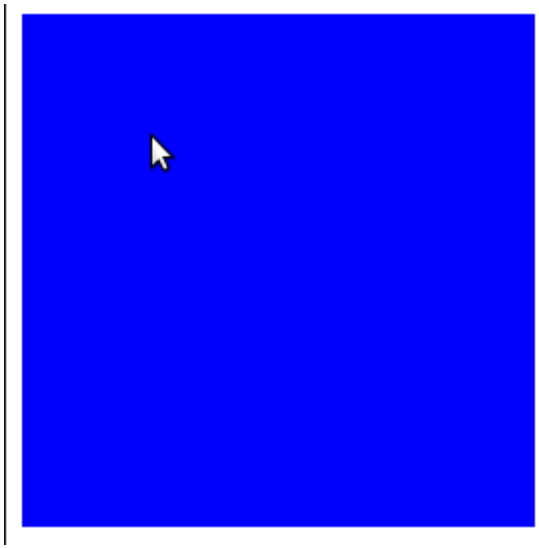


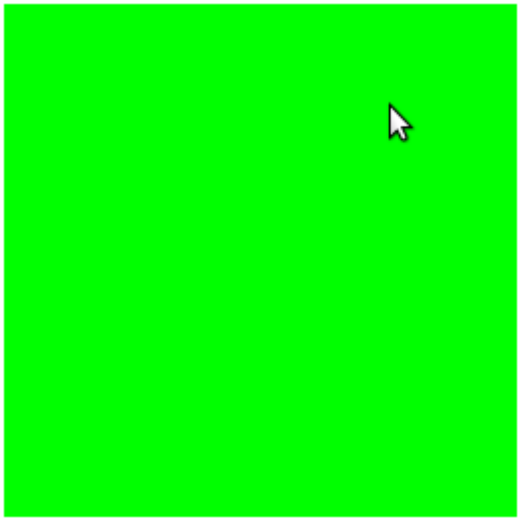
The second one was a bit more complicated, not so much, but a little bit. It was “plain image generator”. I mean, imageplus was generating images as long as the client requested it. The images were in different colors depending on the position the cursor was poining. Thus, if the cursor was in the top left position, the square generated was different from the bottom right position.

I think that this example makes imageplus overwork, because every time the cursor moves, a request is sent to the server, and then imageplus generates an image. If you want to pass from half a square to the other half, how many requests do you need to do? How many images are you generating?

Well, really this is a nonsense without any relevance because I was just testing how imageplus could generate and image and display it through the web.

Here there are a couple of screenshots.





And now the last one. This example consists on an image in which you go with the cursor over it, and you get the red, green and blue average as well as the total average of the neighbor pixels of the pixel you are pointing.

Although I'm going to explain this now, It applies also for the examples above mentioned.
Imageplus uses a type of structure, or better said, a class to store the image, but Wt uses another type of class to store its image. So I needed to connect them somehow.

The Wt's class which can store a image, getting it in a low level, pixel by pixel is called Wt::WRasterImage, then I have done a function which given a image from imageplus, it returns a pointer to WRasterImage filled with this image. This function will be the main connector to transmit images to the web, very useful.
Although Wt can process the pixels by acceding them and changing their value, it is not the goal we are aiming to. We want all the image and video processing to be done with imageplus.
Currently, I'm trying to merge the webpage I did last week with two of the demos, which seems a bit more complex, but not impossible. I'll explain you in the following articles!

Sunday, April 10, 2011

Dynamic web demo of image segmentation

After long time understanding Wt, the C++ library which I have talked about in my last posts, I am glad to release my first web demonstration. This demonstration consists on an image and a slider.

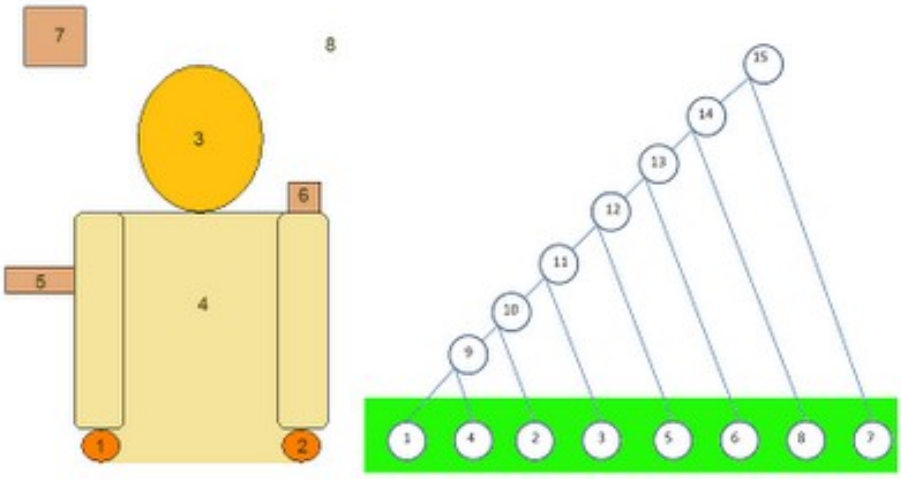
The algorithm I want to demonstrate in this occasion the binary partition tree creation algorithm, an imageplus software that creates a hierarchical partition binary tree with the uniform regions of the image.

Then, the slider at first determines the maxim number of regions the algorithm can segment. This is the 100% of segmentation regions, if the slider is moved down, the number of regions for the segmentation decreases.

In the worst situation, for instance, I had a completely uniform (plain) image, the slider just would have allowed one level, because the minimum is one region, and the maximum is also one region, because is already uniform.

In most images, the diversity of pixel values will allow the definition of several regions. In order to set a maximum value to the slider, the segmentation algorithm is capable of detecting all the plain regions in the image to compute the maximum amount of regions that a the finest possible partition of an image will include.

To know how it works, have a look at this article



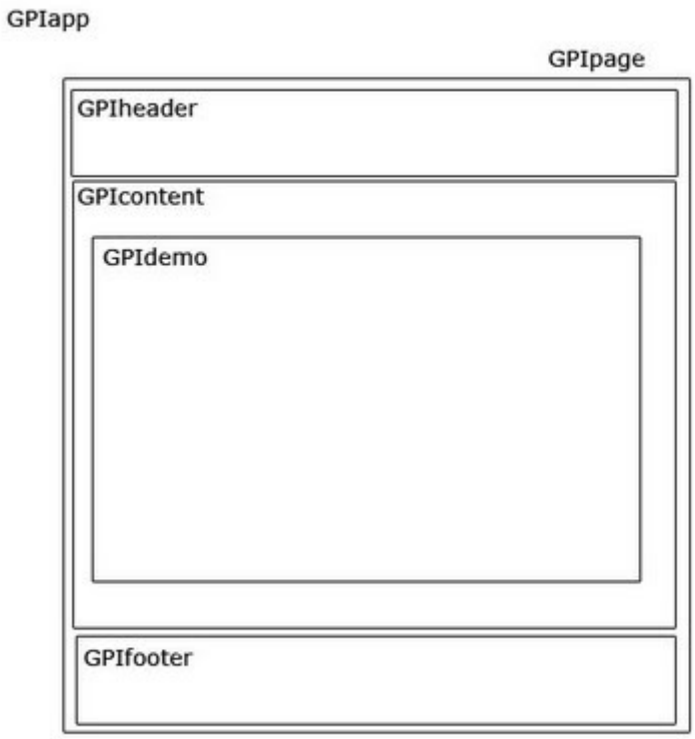
Once the leaves of the tree are computed, the demonstration displays their associated regions by uniformly painting them with their average color, they are called “the leaves of the tree”. The slider then, reflects the number of leaves the tree is going to have.

Here there are a few screen-shots of the demonstration.

Monday, April 11, 2011

Web algorithm demonstrator architecture

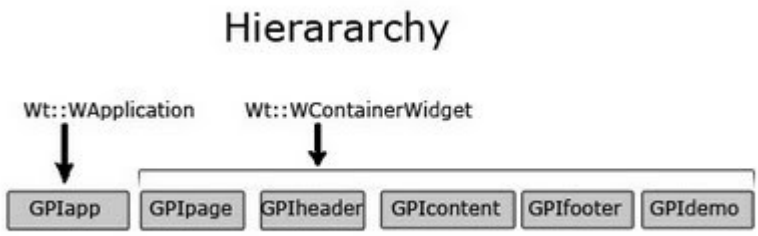
Apart from the demonstration, this week I've been preparing the interface of the web page. It wasn't an easy task. We can talk about two kind of interfaces, the first one, the programmer interface, whose name is “GPIapp”, and is a class which allows programmers to do the most common functionality such as change the title of the demonstration, or set a little description about this algorithm in an easy way.



This interface succeeds in that all the files of the web page I've been working with, classes GPIapp, GPIheader, GPIcontent, GPIfooter and GPIdemo are now transparent for the programmer, because they have just to inherit a class from GPIapp and they have already got the whole web page in the demo, and all they put inside, is going to be placed in the correct position.

This is very important because programmers are able to make web pages very fast and without any knowledge of neither HTML, CSS nor other web technology, and also they are avoided to know too much about Wt.

This is the hierarchy of the new classes. As you can see, almost all the classes inherit from WContainerWidget, this is because this one is the "div" element in HTML, and I needed a framework for my web page.



The second interface, as you know is the graphical interface that you can see when you enter at the web page. The design of this interface is a flexible design. It means that it doesn't matter what resolution or display size you have, the web page is going to fit itself to the screen.

Of course the page allows programmers to put a description as long as they want, and the page is resized to fit that text.

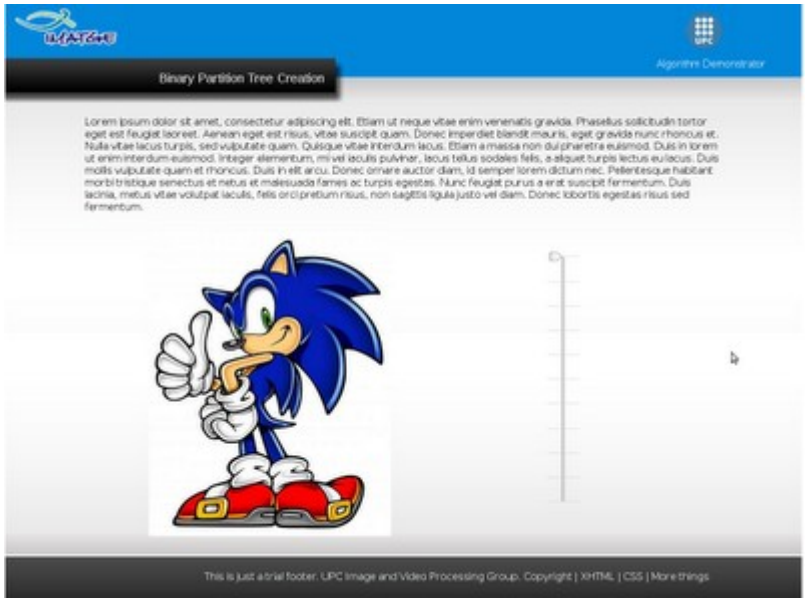


Figure Index